Università di Genova

Dipartimento di
Informatica, Bioingegneria,
Robotica e Ingegneria dei Sistemi

Flexible coinduction

Francesco Dagnino

Theses Series DIBRIS-TH-2021-42

Ph.D. Thesis

Università di Genova
Dipartimento di Informatica, Bioingegneria,
Robotica ed Ingegneria dei Sistemi
Ph.D. Thesis in
Computer Science and System Engineering
Computer Science Curriculum

Flexible coinduction

by

Francesco Dagnino

October 2021

Ph.D. Thesis in Computer Science and System Engineering (S.S.D. INF/01)
Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi
Università di Genova

Candidate
Francesco Dagnino
francesco.dagnino@dibris.unige.it

Title
Flexible coinduction

Advisors
Davide Ancona
DIBRIS, Università di Genova
davide.ancona@unige.it

Elena Zucca
DIBRIS, Università di Genova
elena.zucca@unige.it

External Reviewers
Ugo Dal Lago,
Dipartimento di Informatica - Scienzae Ingegneria, Università di Bologna,
ugo.dallago@unibo.it

Ekaterina Komendantskaya,
School of Mathematics and Computer Science, Heriot-Watt University,
e.komendantskaya@hw.ac.uk

Tarmo Uustalu,
Department of Computer Science, Reykjavik University,
tarmo@ru.is

Location
DIBRIS, Univ. di Genova
Via Opera Pia, 13
I-16145 Genova, Italy

Submitted On
October 2021

Abstract

Recursive definitions of predicates by means of inference rules are ubiquitous in computer science. They are usually interpreted inductively or coinductively, however there are situations where none of these two options provides the expected meaning. In the thesis we propose a flexible form of coinductive interpretation, based on the notion of *corules*, able to deal with such situations.

In the first part, we define such flexible coinductive interpretation as a fixed point of the standard inference operator lying between the least and the greatest one, and we provide several equivalent proof-theoretic semantics, combining well-founded and non-well-founded derivations. This flexible interpretation nicely subsumes standard inductive and coinductive ones and is naturally associated with a proof principle, which smoothly extends the usual coinduction principle.

In the second part, we focus on the problem of modelling infinite behaviour by a big-step operational semantics, which is a paradigmatic example where neither induction nor coinduction provide the desired interpretation. In order to be independent from specific examples, we provide a general, but simple, definition of *what a big-step semantics is*. Then, we extend it to include also *observations*, describing the interaction with the environment, thus providing a richer description of the behaviour of programs. In both settings, we show how corules can be successfully adopted to model infinite behaviour, by providing a construction extending a big-step semantics, which as usual only describes finite computations, to a richer one including infinite computations as well. Finally, relying on these constructions, we provide a proof technique to show soundness of a predicate with respect to a big-step semantics.

In the third part, we face the problem of providing an algorithmic support to corules. To this end, we consider the restriction of the flexible coinductive interpretation to *regular* derivations, analysing again both proof-theoretic and fixed point semantics and developing proof techniques. Furthermore, we show that this flexible regular interpretation can be equivalently characterised inductively by a *cycle detection* mechanism, thus obtaining a sound and complete (abstract) (semi-)algorithm to check whether a judgement is derivable. Finally, we apply such results to extend logic programming by *coclauses*, the analogous of corules, defining declarative and operational semantics and proving that the latter is sound and complete with respect to the regular declarative model, thus obtaining a concrete support to flexible coinduction.

Acknowledgements

I am very grateful to my supervisors, Elena Zucca and Davide Ancona. Their advice and guidance have been essential for the development of the results of this thesis. They have always encouraged me to develop my ideas, but helping me to make them clear and concrete through several, extreemely useful, discussions, for which they have always been available. Basically, from them I have learnt how to do researc and my growth in these years would not have been possible without them.

A special thank goes to Pino Rosolini, for sure one of the best teacher I have ever met. He and his wonderful classes not only have taught me the large part of what I know about mathematical logic and category theory, but they have given me the opportunity to develop a way of thinking, precise and rigorous, which has been essential for my growth as a researcher.

I am also greateful to all the coauthors who have worked with me in these years, Jurriaan Rot, Mariangiola Dezani and Viviana Bono. Their contribution has been very important to achieve several results in this thesis.

I also would like to thank Ugo Dal Lago, Ekaterina Komendantskaya and Tarmo Uustalu for having reviewed this thesis in the short amount of time they have had at their disposal. Their comments have been very helpful to improve the presentation.

I am very grateful to my family, especially to my parents Monica and Stefano and my brother Federico. They have always supported me with willingness and patience and encoraged to persue my goals. I would not be here without them.

Special thanks go to my friends Paolo, Federico and Daniele. Their presence and the countless discussions about everything, from the “chief systems” to everyday life, have been of great importance for me over these years.

Last but not least, in the last three years I shared wonderful moments with many colleagues and friends. Thank you all. Among them, special thanks go to Carola, Enrico, Federico, Francesco, Luca, Matelda, Pietro and Veronica, and to all friends of the room 309.

Contents

Introduction

1

Introduction

Inference systems are a versatile and widely used framework to define and reason about possibly recursive predicates, such as small-step and big-step operational semantics, type systems, sequent calculi and other proof systems. The key feature of inference systems is that they express definitions by means of *(inference) rules*, which are if-then clauses, making explicit the steps we can and have to do to prove judgements.

They support both inductive and coinductive reasoning in a pretty natural way: in inductive reasoning we are only allowed to use finite derivations, while in the coinductive one we can prove judgements by arbitrary, finite or infinite, derivations. Furthermore, in both cases we have proof principles, the induction and the coinduction principle, to reason about defined judgements.

These two interpretations of a set of rules are very different from each other. The inductive interpretation is the smallest one, as it is restricted only to finite derivations, but, in return, it implicitly provides us with an (abstract) algorithm[1], which looks for a finite derivation of a judgement; such an algorithm is sound and complete with respect to derivable judgements. That is, it may not terminate for judgements that do not have a finite derivation, but it is guaranteed to successfully terminate, finding a finite derivation, for all and only derivable judgments. Instead, the coinductive interpretation is the largest one, as it allows any, finite or not, derivations, but there is no hope, in general, to find an algorithm which successfully terminates for derivable judgments. The reason, intuitively, is that there can be derivations requiring *infinitely many* different judgements to be proved.

This strong dichotomy between inductive and coinductive interpretation makes the framework of inference systems a bit rigid. Indeed, it allows us to choose only between two possibilities, while there are cases where neither the inductive nor the coinductive interpretation are able to provide the expected meaning, as it lies between these two extremes. Let us illustrate this fact by a paradigmatic example: the definition of big-step operational semantics explicitly modelling divergence. We consider the standard call-by-value λ-calculus. The big-step judgement has shape $e \Rightarrow v$, meaning that expression e evaluates to value v. Below are the standard rules for the big-step operational

1 Here and throughout the thesis, we use the word "algorithm" to indicate a procedure which is not required to terminate.

semantics:

$$\frac{}{v \Rightarrow v} \qquad \frac{e_1 \Rightarrow \lambda x.e \quad e_2 \Rightarrow v_2 \quad e[v_2/x] \Rightarrow v}{e_1\ e_2 \Rightarrow v}$$

These rules clearly model only converging computations, namely, computations returning a value. To take into account divergence, we can add a special result ∞ and rules handling it:

$$\frac{e_1 \Rightarrow \infty}{e_1\ e_2 \Rightarrow \infty} \qquad \frac{e_1 \Rightarrow \lambda x.e \quad e_2 \Rightarrow \infty}{e_1\ e_2 \Rightarrow \infty}$$

$$\frac{e_1 \Rightarrow \lambda x.e \quad e_2 \Rightarrow v_2 \quad e[v_2/x] \Rightarrow \infty}{e_1\ e_2 \Rightarrow \infty}$$

The intuition behind these rules is that as soon as a premise diverges, the conclusion should diverge as well. Now the question is the following: how should we interpret the whole set of rules? Clearly, if we take the inductive interpretation, we cannot derive any judgement for diverging expressions, as this interpretation only allows finite derivations and there is no axiom (rule with no premises) introducing divergence (∞). On the other hand, the coinductive interpretation allows the derivation of too many judgements for diverging expressions. For instance, if we consider autoapplication $\Omega = \omega\,\omega$, with $\omega = \lambda x.x\,x$, then we have the following infinite derivation, which is correct for any v_∞, denoting either a value or ∞:

$$\frac{\dfrac{}{\omega \Rightarrow \omega} \quad \dfrac{}{\omega \Rightarrow \omega} \quad \dfrac{\vdots}{\Omega \Rightarrow v_\infty}}{\Omega \Rightarrow v_\infty}$$

while the only expected judgement is $\Omega \Rightarrow \infty$.

Hence, none of the two standard interpretations of inference systems is capable to provide the intending meaning. In this thesis we tackle this problem, introducing a generalisation of inference systems, providing more flexibility when choosing the interpretation of the given set of rules. We call this approach *flexible coinduction* as it allows us to refine the coinductive interpretation.

The key concept of the proposed generalisation are *corules*, which are special rules that need to be provided together with standard rules, and are used to tune their semantics. More precisely, they allow us to disregard some undesired infinite derivations, thus obtaining an interpretation which is not necessarily either the smallest (inductive) or the largest (coinductive) one. For instance, in the above example, the coinductive interpretation is undetermined on diverging expressions (we can derive both correct and incorrect judgements); but, as we will see, adding suitable corules we can remove all incorrect judgements, thus obtaining the correct interpretation. An important property is that standard inductive and coinductive interpretations are particular cases, that is, they can be recovered by specific choices of corules, thus this framework indeed generalises standard inference systems.

The thesis starts by studying inference systems with corules in their general properties (Part I). Nicely, all standard notions and results about inference

systems (fixed point constructions, model-theoretic and proof-theoretic semantics, their equivalence and associated proof techniques) smoothly extend to this generalised setting, providing solid and fairly simple foundations to flexible coinduction. After this general study, we deepen the analysis of flexible coinduction in two directions: on one side, we address the above mentioned paradigmatic example of big-step semantics modelling also infinite behaviour (Part II), on the other one, we face the problem of providing a concrete algorithmic support to flexible coinduction (Part III).

In the former direction, we show how corules can be successfully adopted to define big-step semantics modelling also infinite behaviour. We consider first semantic descriptions, like the one sketched above, where the behaviour of the program is just described by its final result, if any, and ∞ in case of divergence. Then, we extend the approach to more complex descriptions where, in addition to the final result, we also have *observations*, modelling the interaction with the environment (*e.g.*, traces of events, memory usage, costs etc.). In this latter case, considering also infinite behaviour is even more challenging, as we need to model possibly infinite observable interactions. The key contribution is that, rather than studying big-step semantics on example languages, we take a general perspective, developing our definitions and results for an arbitrary big-step semantics, abstracting from specific features of concrete instances. The generality of our approach is witnessed by a broad class of examples.

In the latter direction, as previously noticed, even for the standard coinductive interpretation, in general, there is no complete algorithm which looks for a derivation, so the same holds for our generalised framework, as it subsumes standard coinduction. Therefore, to provide an algorithmic support to corules, we need to consider a restriction of the general model. As it is customary in standard coinduction, we consider the restriction to *regular* derivations, that is, derivations involving only finitely many different judgements. All notions and results discussed in the general setting can be smoothly adapted to the regular setting, thus providing solid foundations also to flexible regular coinduction. From the algorithmic perspective, the interesting result is that flexible regular coinduction has an equivalent inductive characterisation, which, as previously mentioned, provides us with a sound and complete (abstract) algorithm to find a derivation. Building on this general analysis, we define an extension of logic programming supporting flexible coinduction[2], restricted to the regular case, like standard coinductive logic programming, thus providing a concrete executable support to our general framework.

1.1 Outline

PART I We present the framework of inference systems with corules, an extension of standard inference systems supporting a flexible form of coinduction.

2 A prototype SWI-Prolog implementation is available at https://github.com/davideancona/coLP-with-coclauses.

CHAPTER 2 We present background notions on standard inference systems in full detail: proof-theoretic and model-theoretic semantics, their equivalence, and associated proof techniques.

CHAPTER 3 We present inference systems with corules and their general properties, which smoothly extend standard results for inference systems. We define a model-theoretic semantics in terms of fixed points, several equivalent proof-theoretic semantics, and associated proof-techniques.

CHAPTER 4 We discuss related work and outline directions for future work.

PART II We study big-step operational semantics, analysing how it can be used to model and reason about infinite behaviour of programs. To address this problem, flexible coinduction can be successfully used to get precise semantic models. From a methodological point of view, we do not work on specific examples, but, rather, we take an abstract perspective. That is, we provide a general definition of big-step semantics, which as usual only models finite behaviour, then we define computations by means of a transition relation driven by rules. Finally, we define constructions that extend a given big-step semantics to take into account infinite behaviour as well, proving their correctness against the previously introduced notion of computation. This provides us with a self-contained coherent treatment of big-step semantics, independent from other approaches.

CHAPTER 5 We focus on standard big-step semantics, defining various constructions distinguishing stuck from diverging computations, where corules play a crucial role. Further, relying on such constructions, we express soundness of a predicate against a big-step semantics, and describe a proof technique to show such property, proving its correctness.

CHAPTER 6 We extend the notion of big-step semantics of the previous chapter, to take into account the observable behaviour of a program during a computation, represented by an element of a given monoid of finite observations. We then define a completion construction from monoids to *ω-monoids*, an algebraic structure used to model possibly infinite observations. Finally, using corules, we extend a given big-step semantics with observations to model infinite computations, with their possibly infinite observable behaviour, as well.

CHAPTER 7 We discuss related work and outline directions for future work.

PART III We consider the restriction of coinduction to regular derivations, extending results about flexible coinduction to this restricted setting. Then, we apply these notions to define an extension of coinductive logic programming supporting flexible coinduction.

CHAPTER 8 We study the regular interpretation of inference systems, defining the proof-theoretic semantics, in terms of regular derivations,

the model-theoretic semantics, as an instance of the rational fixed point, and an equivalent inductive characterisation. We discuss associated proof techniques. Further, we extend all these results to inference systems with corules, thus defining flexible regular coinduction.

CHAPTER 9 We present flexible coinductive logic programming, which is the logic programming counterpart of inference systems with corules. We define its declarative and operational semantics, proving the latter is sound and complete with respect to the regular restriction of the declarative semantics.

CHAPTER 10 We discuss related work and outline directions for future work.

1.2 Relationship with published and submitted papers

The content of Part I, in particular Chapter 3, originates from a work published at *ESOP 2017* (Ancona, Dagnino, and Zucca, 2017b) and subsequently extended by a paper on *LMCS* (Dagnino, 2019). Differently from these papers, which are mainly focused on coaxioms, here we present directly the more general framework of inference system with corules, as it is the one we need in the rest of the thesis, and omit some technical results, which are not needed.[3]

The content of Part II originates from two papers published at *OOPSLA 2017* (Ancona, Dagnino, and Zucca, 2017c) and *ECOOP 2018* (Ancona, Dagnino, and Zucca, 2018), where we analyse examples of big-step semantics, showing how corules can be successfully adopted to model also infinite behaviour. In Part II we take a more abstract and systematic approach, outlined in an *ICTCS 2018* paper (Dagnino, 2018) and developed in an *ESOP 2020* paper (Dagnino et al., 2020) and an *SCP* paper (Ancona et al., 2020a), in a special issue of *ECOOP 2020*. That is, rather than considering specific examples, we provide a general definition of big-step semantics, which as usual only considers finite behaviour, and define constructions extending a given big-step semantics to model infinite behaviour as well. With respect to the *ESOP* paper, Chapter 5 focuses more on big-step semantics in itself, rather than on the proof technique for soundness, which is presented as an important application of the discussed approach. Further, Chapter 5 considers a more general notion of big-step semantics (cf. Definition 5.1), which is closer to concrete examples, and covers a broader class of them. In addition, we define a construction based on corules, which generalises examples in the OOPSLA paper. In the SCP paper, to prove the correctness of the construction, we prove the equivalence of the resulting big-step semantics with respect to a reference small-step semantics; differently, in Chapter 6, we follow the approach of Chapter 5 and of the ESOP paper, showing the correctness with respect to a transition system derived from big-step rules. In this way, again the definition of big-step semantics with

3 We refer to the LMCS paper (Dagnino, 2019) for them.

observations (cf. Definition 6.21) is more general, the approach is more uniform and proofs become simpler. The comparison with a small-step semantics, which can be found in the SCP paper, is omitted.

The content of Part III is taken from a paper submitted to *LMCS* (Dagnino, 2020), for Chapter 8, and from a paper presented at *ICLP 2020* and published in *TPLP* (Dagnino, Ancona, and Zucca, 2020), which extends a preliminary work published in the post-proceedings of *CoALP-Ty 2016* (Ancona, Dagnino, and Zucca, 2017a), for Chapter 9. Related results, focused on object-oriented programming, and not included in the thesis, can be found in papers presented at *ICTCS 2019* (Barbieri et al., 2019), *ECOOP 2020* (Ancona et al., 2020b) and *FTfJP 2020* (Barbieri, Dagnino, and Zucca, 2020).

1.3 Notations

Given a set X, we denote by $\wp(X)$ the power-set of X, that is, the set of all subsets of X, and by $\wp_\omega(X)$ the finite power-set of X, that is, the set of all finite subsets of X. Given a function $f : X \to Y$, we denote by $f_! : \wp(X) \to \wp(Y)$ the direct image of f, that is, for all $A \subseteq X$, $f_!(A) = \{y \in Y \mid y = f(x) \text{ for some } x \in A\}$, and by $f^* : \wp(Y) \to \wp(X)$ the inverse image of f, that is, for all $B \subseteq Y$, $f^*(B) = \{x \in X \mid f(x) \in B\}$.

Given a set X, we denote by $X^\star$, X^ω, and $X^\infty = X^\star + X^\omega$, respectively, the sets of finite, infinite, and possibly infinite sequences of elements of X. Infinite sequences on X, namely, elements of X^ω, are often identified with functions of type $\mathbb{N} \to X$. We write $x{:}u$ for concatenation of $x \in X$ with $u \in X^\infty$, $u \cdot v$ for concatenation of $u \in X^\star$ with $v \in X^\infty$, and ε for the empty sequence. We will often omit : and $\cdot$ when clear form the context. Given a function $f : X \to Y$, we obtain functions $f^\star : X^\star \to Y^\star$, $f^\omega : X^\omega \to Y^\omega$ and $f^\infty : X^\infty \to Y^\infty$ defined by elementwise application of f. For $u \in X^\star$ and $v \in X^\infty$, we say that u is a prefix of v, if $u \cdot z = v$ for some $z \in X^\infty$.

PART I

Flexible coinductive definitions

2

Inference systems

Inference systems are a widely used framework to define and reason about several kinds of judgements by means of *(inference) rules*. Each rule specifies an if-then condition, saying that a certain judgement holds provided that some other judgements hold as well.

Inference systems support both inductive and coinductive reasoning in a pretty natural way: in inductive reasoning we are only allowed to use finite derivations, while in the coinductive one we can prove judgements by arbitrary, finite or infinite, derivations. Furthermore, in both cases we have proof principles, the induction and the coinduction principles, to reason about defined judgements.

As inference systems will be used throughout the whole thesis, in this chapter we provide all the background notions needed in the rest of the thesis. Section 2.1 introduces inference systems and defines their semantics in proof-theoretic style. [1] Section 2.2 reports results about fixed points of functions on complete lattices, we use in Section 2.3, to define a model-theoretic semantics of inference systems, proving its equivalence with the proof-theoretic one. In Section 2.4 we describe proof techniques, notably, the induction and coinduction principles, and, finally, in Section 2.5, we discuss iterative characterisation of inductive and coinductive semantics of inference systems.

All results we present are pretty well-known, we refer to works by Aczel (1977), Leroy and Grall (2009), and Sangiorgi (2011), however, especially the proof-theoretic semantics is not discussed in a sufficiently rigorous way. Hence, we provide all the necessary details about trees (cf. Section 2.1.1) to develop such a proof-theoretic semantics, and carry out a new, as far as we know, proof of equivalence between such proof-theoretic semantics and the model-theoretic one, expressed in terms of fixed point. This proof relies on a general framework to relate these two styles for defining semantics of inference systems, based on an adjunction (cf. Section 2.3). This rigorous development will be essential in next chapters to prove similar equivalence results for interpretations going beyond standard induction and coinduction.

1 In this thesis by proof-theoretic semantics we mean the semantics of inference systems expressed in terms of proof trees, as done by Leroy and Grall (2009).

2.1 Inference systems and proof trees

In this section we introduce inference systems and their inductive and coinductive interpretation in proof-theoretic style (Aczel, 1977; Sangiorgi, 2011). Let us assume a *universe* $\mathcal{U}$, which is a set whose elements are called *judgements*, ranged over by j.

DEFINITION 2.1 : An *inference rule*, or simply a *rule*, is a pair $\langle Pr, c\rangle$ where $Pr \subseteq \mathcal{U}$ is the set of *premises* and $c \in \mathcal{U}$ is the *conclusion* (a.k.a. *consequence*). A rule with an empty set of premises is an *axiom*. An *inference system* $\mathcal{I}$ is a set of rules.

Intuitively, a rule states an if-then condition on judgements: if the premises hold, then the conclusion should hold as well, hence an axiom requires a judgement to hold without any precondition. In the following, as it is customary, we will often write a rule $\langle Pr, c\rangle$ using the fraction notation, that is, $\dfrac{Pr}{c}$. A set of rules, that is, an inference system, defines a set of *derivable* judgements. There are several ways to choose this set, but it has to satisfy some properties with respect to the inference system:

DEFINITION 2.2 : Let $\mathcal{I}$ be an inference system and $S \subseteq \mathcal{U}$ a set of judgements. We say that

- S is *$\mathcal{I}$-closed* if, for all rules $\langle Pr, c\rangle \in \mathcal{I}$, if $Pr \subseteq S$ then $c \in S$,
- S is *$\mathcal{I}$-consistent* if, for all $j \in S$, there is a rule $\langle Pr, c\rangle \in \mathcal{I}$ with $c = j$ and $Pr \subseteq S$,
- S is an *$\mathcal{I}$-interpretation* if it is $\mathcal{I}$-closed and $\mathcal{I}$-consistent.

In the following we will omit the reference to the inference system when clear from the context. Intuitively, rules can be used to derive judgements from a set of given judgements. The definition of interpretation requires a kind of "stability" condition with respect to rules: if S is an interpretation, all judgements that can be derived from S are already in S (S is closed), and all judgements in S can be derived by judgements in S (S is consistent).

REMARK: The definition of inference systems is purely semantic. This allows us to develop the theory in an abstract way, independently from a specific syntax.[2] However, typically an inference system consists of *infinitely many* rules, so it is not possible to write down all rules in an extensional way. Hence, as it is common practice, in the examples throughout this thesis we describe inference systems by means of *meta-rules* or *rule schemes*. Meta-rules describe all possible shapes that rules can assume, using some syntax with (meta-)variables to range over base elements. Then, the concrete inference system can be easily recovered by instantiating variables with all their possible values.

2 Note that considering a specific syntax is quite straightforward, for instance as we do in Chapter 9 in the context of logic programming.

Let us show some examples to illustrate this concept. We denote by $\mathbb{Z}$ the set of integers and by $\mathbb{Z}^\star$ the set of finite lists (sequences) of integers. We consider the definition of the predicate $\mathsf{member}(x, l)$, that holds if the element x occurs in the list l. In this case the universe can be the set $\{\mathsf{member}(x, l) \mid x \in \mathbb{Z}, l \in \mathbb{Z}^\star\}$, so, for instance, judgements like $\mathsf{member}(1, \varepsilon)$, $\mathsf{member}(3, 1{:}3{:}\varepsilon)$ or $\mathsf{member}(1, 1{:}3{:}2{:}\varepsilon)$ are in the universe. Therefore, the definition of the judgement $\mathsf{member}(x, l)$ through an inference system looks like the following:

$$\frac{}{\mathsf{member}(x, x{:}l)} \qquad \frac{\mathsf{member}(x, l)}{\mathsf{member}(x, y{:}l)}$$

where $x, y \in \mathbb{Z}$ and $l \in \mathbb{Z}^\star$. Actual rules can be obtained from these schemes by instantiating variables with all their possible values.

Another example is the judgement $\mathsf{allPos}(l)$, that holds if all elements in l are strictly positive integers. The universe in this case can be $\{\mathsf{allPos}(l) \mid l \in \mathbb{Z}^\star\}$ and the definition as inference system is the following:

$$\frac{}{\mathsf{allPos}(\varepsilon)} \qquad \frac{\mathsf{allPos}(l)}{\mathsf{allPos}(x{:}l)}\ x > 0$$

This example shows another important feature of meta-rules: *side conditions*. Beside the second meta-rule, we have specified a predicate ($x > 0$), that x must satisfy. In general, side conditions are predicates on variables occurring in the meta-rule, restricting the set of values on which variables range over, thus reducing the set of instances of the meta-rule. They are extremely useful to provide a finer control on instances of rule schemes, and without them many definitions would be very difficult to express as inference systems. For instance, the definition of $\mathsf{allPos}(l)$ without side conditions reported below requires an additional judgement $\mathsf{pos}(x)$, that holds if x is strictly positive.

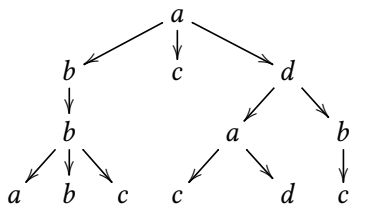
$$\frac{}{\mathsf{pos}(1)} \qquad \frac{\mathsf{pos}(x)}{\mathsf{pos}(x+1)} \qquad \frac{}{\mathsf{allPos}(\varepsilon)} \qquad \frac{\mathsf{pos}(x) \quad \mathsf{allPos}(l)}{\mathsf{allPos}(x{:}l)}$$

Until now, we have focused on how to write down definitions through inference systems, relying on readers' intuition that a given inference system actually defines an intended predicate. In order to formally prove that such definitions correctly capture their intended meaning, we need to define in a rigorous way how an inference system can be interpreted. More precisely, given an inference system $\mathcal{I}$, we have to assign to it an $\mathcal{I}$-interpretation (see Definition 2.2), which will be its semantics.

We first address this issue in a *proof-theoretic* style, that allows us to define a very intuitive semantics of inference systems. This semantics is based on the notion of *proof tree* or *derivation*, that is, a tree where every node is (labelled by) the conclusion of a rule and its children nodes are (labelled by) the premises of such rule. To make this definition precise, in the next section we introduce a class of trees with the properties needed to define and reason about derivations.

2.1.1 A digression on trees

Here we report some definitions and results about trees. We essentially follow the approach adopted by Courcelle (1983), Aczel, Adámek, and Velebil (2001),



$\mathsf{r} = a$
$L = \{\varepsilon,$
$b, c, d,$
$bb, da, db,$
$bba, bbb, bbc, dac, dad, dbc\}$

FIGURE 2.1 An example of tree on $\{a, b, c, d\}$.

Aczel et al. (2003), Moerdijk and Palmgren (2000), van den Berg and De Marchi (2007), and Adámek et al. (2015), with some differences in the definition of tree, due to the specific context where we use trees. We start by some preliminary definitions.

Given a set A, we denote by $A^\star$ the set of finite sequences on the alphabet A. We denote by ε the empty sequence and, given $\alpha, \beta \in A^\star$, we denote by juxtaposition $\alpha\beta$ their concatenation. A *tree language* on a set A is a non-empty and prefix-closed subset $L \subseteq A^\star$, that is, such that, for all $\alpha \in A^\star$ and $x \in A$, if $\alpha x \in L$ then $\alpha \in L$. Hence, in particular, the empty sequence belongs to any tree language.

DEFINITION 2.3 : A *tree* τ on a set A is a pair $\langle r, L\rangle$ where L is a tree language on A and $r \in A$ is the *root* of the tree. We set $\mathsf{N}(\tau) = L$ and $\mathsf{r}(\tau) = r$.

Intuitively, a sequence $\alpha \in L$ represents a node of the tree labelled by $\tau(\alpha)$, which is defined as follows:

$$\tau(\alpha) = \begin{cases} \mathsf{r}(\tau) & \alpha = \varepsilon \\ x & \alpha = \beta x \end{cases}$$

Therefore, a tree τ on A induces a partial function from $A^\star$ to A whose domain is a tree language. Differently from the literature[3] (Courcelle, 1983; Aczel et al., 2003), Definition 2.3 forces trees to be unordered and, more importantly, it ensures that there cannot be two sibling nodes with the same label. These two additional requirements are reasonable in our setting, as we will use trees to define derivations, where sibling nodes correspond to premises of a rule which are a set, hence unordered and with no repetitions. Furthermore, these requirements will turn out to be essential in the proof of the main result of this section, namely, Theorem 2.4. In Figure 2.1 we report an example of tree with labels in $\{a, b, c, d\}$ represented according to our definition.

Given a tree τ and a node $\alpha \in \mathsf{N}(\tau)$, we denote by $\tau_{|_\alpha}$ the *subtree of* τ *rooted at* α, defined as the pair $\langle \tau(\alpha), \{\beta \in A^\star \mid \alpha\beta \in \mathsf{N}(\tau)\}\rangle$ and denote by $\mathsf{SubTr}(\tau)$ the set of all subtrees of τ. We also define $\mathsf{chl}_\tau(\alpha) = \{\tau_{|_\beta} \mid \exists x \in A.\beta = \alpha x, \beta \in \mathsf{N}(\tau)\}$ the set of *children* of α in τ and $\mathsf{dst}(\tau) = \mathsf{chl}_\tau(\varepsilon)$ the set of *direct subtrees* of τ, which are the children of the root of τ. Note that, for all $\alpha \in \mathsf{N}(\tau)$, we have $\tau(\alpha) = \mathsf{r}(\tau_{|_\alpha})$ and $\mathsf{chl}_\tau(\alpha) = \mathsf{dst}(\tau_{|_\alpha})$. We will write $\tau' \prec \tau$ iff $\tau' \in \mathsf{dst}(\tau)$, that is, τ' is a direct subtree of τ. A tree τ is *well-founded* iff the relation $\prec$

3 We refer to (Dagnino, 2019) for a detailed comparison.

restricted to $\mathsf{SubTr}(\tau)$ is well-founded, namely, there are no infinite chains in $\prec$. Intuitively, this means that in τ there are no infinite paths. It is easy to check that τ is well-founded iff all $\tau' \in \mathsf{dst}(\tau)$ are well-founded.

In the following, assume a set A and denote by $\mathcal{T}_A$ the set of all trees on A. The main theorem of this subsection (Theorem 2.4) is inspired by results presented by Aczel et al. (2003) and Adámek et al. (2015), even though they need a different definition of trees, since they are focused on different properties.[4] This result is essential to provide the fixed point characterisation of the coinductive interpretation of inference systems (cf. Theorem 2.24). We show that, starting from a graph structure on a subset of A, for each node of the graph there is a unique way to construct a tree on A coherent with the graph structure. In this context a graph is a function $g : X \to \wp(X)$, modelling the adjacency function, that is, X is the set of nodes and, for all $x \in X$, $g(x)$ is the set of adjacents of x.

THEOREM 2.4 : Let $g : X \to \wp(X)$ be a function and $v : X \to A$ be an injective function. Then, there exists a unique function $p : X \to \mathcal{T}_A$ such that the following diagram commutes:

$$\begin{array}{ccc} X & \xrightarrow{\quad p \quad} & \mathcal{T}_A \\ {\scriptstyle \langle g, v \rangle}\downarrow & & \downarrow{\scriptstyle \langle \mathsf{dst}, \mathsf{r} \rangle} \\ \wp(X) \times A & \xrightarrow{p_! \times \mathrm{id}_A} & \wp(\mathcal{T}_A) \times A \end{array}$$

moreover, p is injective.

Proof: For all $x \in X$, we define the set $L_{x,n}$ of paths of length n starting from x and the set L_x of all paths starting from x as follows:

$$L_x = \bigcup_{n \in \mathbb{N}} L_{x,n} \qquad \begin{aligned} L_{x,0} &= \{\varepsilon\} \\ L_{x,n+1} &= \bigcup_{y \in g(x)} \{v(y)\alpha \mid \alpha \in L_{y,n}\} \end{aligned}$$

Trivially we have, for all $x \in X$, $L_x \subseteq A^\star$. We show, by induction on n, that for all $n \in \mathbb{N}$, $x \in X$, $\alpha \in A^\star$ and $a \in A$, if $\alpha a \in L_{x,n+1}$ then $\alpha \in L_{x,n}$.

Case: 0 Since $\alpha a \in L_{x,1}$, we have $\alpha = \varepsilon \in L_{x,0}$, as needed.

Case: $n+1$ Since $\alpha a \in L_{x,n+2}$, by definition of $L_{x,n+2}$, we have $\alpha = v(y)\beta$, for some $y \in g(x)$, and $\beta a \in L_{y,n+1}$. By induction hypothesis, we get $\beta \in L_{y,n}$, then, by definition of $L_{x,n+1}$, we get $\alpha = v(y)\beta \in L_{x,n+1}$, as needed.

This implies that L_x is prefix-closed, thus a tree language, and so $\langle x, L_x \rangle$ is a tree on A. We define $p(x) = \langle x, L_x \rangle$.

To prove that the diagram commutes, we have to show that, for all $x \in X$ and $\tau \in \mathcal{T}_A$, $\mathsf{r}(p(x)) = v(x)$, which is true by construction of p, and $\tau \in \mathsf{dst}(p(x))$ iff $\tau = p(y)$ for some $y \in g(x)$. First of all, note that, for all $y \in g(x)$ and $\alpha \in A^\star$, we have $v(y)\alpha \in L_x$ iff $\alpha \in L_y$: if $\alpha \in L_y$ then $\alpha \in L_{y,n}$, for some

4 They want to define a final coalgebra for suitable functors.

$n \in \mathbb{N}$, thus $v(y)\alpha \in L_{x,n+1} \subseteq L_x$, and, if $v(y)\alpha \in L_x$ then $v(y)\alpha \in L_{x,n+1}$, for some $n \in \mathbb{N}$, thus there is $z \in g(x)$ such that $v(z) = v(y)$ and $\alpha \in L_{z,n} \subseteq L_z$, but, since v is injective, we get $z = y$ and so $\alpha \in L_y$. From this fact we immediately get that $p(y) \in \mathsf{dst}(p(x))$, for all $y \in g(x)$. On the other hand, if $\tau \in \mathsf{dst}(p(x))$, then $\tau = p(x)_{|a}$, for some $a \in A$, that is, $\tau = \langle a, \{\alpha \in A^\star \mid a\alpha \in L_x\}\rangle$. In particular, we have $a \in L_{x,1} \subseteq L_x$, hence $a = v(y)$, for some $y \in g(x)$. Therefore, again thanks to the fact above, we get $\tau = \langle v(y), L_y\rangle = p(y)$, as needed.

To prove uniqueness, consider a function $q : X \to \mathcal{T}_A$ making the diagram commute. Then, $\mathsf{r}(q(x)) = v(x) = \mathsf{r}(p(x))$, hence we have only to show that $\mathsf{N}(q(x)) = L_x$. Therefore, we prove by induction on $\alpha \in A^\star$ that, for all $x \in X$, $\alpha \in \mathsf{N}(q(x))$ iff $\alpha \in L_x$.

Case: ε The thesis is trivial.

Case: $a\alpha$ We have $a\alpha \in \mathsf{N}(q(x))$ iff $\alpha \in \mathsf{N}(q(x)_{|a})$ and, since the diagram commutes, hence $q(x)_{|q} = q(y)$, for some $y \in g(x)$, this is equivalent to $a = \mathsf{r}(q(x)_{|a}) = \mathsf{r}(q(y)) = v(y)$ and $\alpha \in \mathsf{N}(q(y))$, for some $y \in g(x)$. By induction hypothesis, this is equivalent to $a = v(y)$ and $\alpha \in L_y$, which is equivalent to $a\alpha \in L_x$.

Finally, we note that p is injective: if $p(x) = p(y)$ then $v(x) = \mathsf{r}(p(x)) = \mathsf{r}(p(y)) = v(y)$, hence $x = y$ because v is injective. □

2.1.2 A proof-theoretic semantics

In this section we define the semantics of inference systems in *proof-theoretic* style. This means that we will assign to an inference system a set of judgements for which we can construct an object that would be the witness of the "truth" of the judgement. These objects are named *proof trees* or *derivations* and are defined below:

DEFINITION 2.5 : Let $\mathcal{I}$ be an inference system, a *proof tree* (or *derivation*) in $\mathcal{I}$ is a tree τ on $\mathcal{U}$, such that, for each node $\alpha \in \mathsf{N}(\tau)$, we have $\langle \mathsf{r}_!(\mathsf{chl}_\tau(\alpha)), \tau(\alpha)\rangle \in \mathcal{I}$.

In other words, a proof tree in $\mathcal{I}$ is a tree τ on the universe $\mathcal{U}$, where, for each node labelled by c whose children nodes are labelled by Pr, the rule $\langle Pr, c\rangle$ belongs to $\mathcal{I}$. Note that the fact that children of a node are unordered and have distinct labels is essential to get a one-one correspondence with the set of premises of a rule.

In the following, we will often represent proof trees using stacks of rules, that is, if $\langle Pr, c\rangle \in \mathcal{I}$ and $T = \{\tau_i \mid i \in I\}$ is a collection of trees such that $\mathsf{r}_!(T) = Pr$ and $\mathsf{r}(\tau_i) = \mathsf{r}(\tau_j)$ implies $i = j$, then we denote by $\dfrac{T}{c}$ the proof tree $\tau = \langle c, \mathsf{N}(\tau)\rangle$ where

$$\mathsf{N}(\tau) = \{\varepsilon\} \cup \bigcup_{i \in I} \mathsf{r}(\tau_i)\mathsf{N}(\tau_i)$$

We say that a tree τ is a proof tree for a judgement $j \in \mathcal{U}$ if it is a proof tree rooted in j, that is, $\mathsf{r}(\tau) = j$. With this terminology we can define two interpretations of an inference system:

DEFINITION 2.6 : Let $\mathcal{I}$ be an inference system:

- the *inductive interpretation* of $\mathcal{I}$, denoted by $\mu[\![\mathcal{I}]\!]$, is the set of judgements having a well-founded proof tree, and
- the *coinductive interpretation* of $\mathcal{I}$, denoted by $\nu[\![\mathcal{I}]\!]$, is the set of judgements having an arbitrary (well-founded or not) proof tree.

We will write $\mathcal{I} \vdash_\mu j$ for $j \in \mu[\![\mathcal{I}]\!]$ and $\mathcal{I} \vdash_\nu j$ for $j \in \nu[\![\mathcal{I}]\!]$. Clearly, by definition, $\mu[\![\mathcal{I}]\!] \subseteq \nu[\![\mathcal{I}]\!]$, but the converse is not necessarily true; indeed when the two interpretations are equal we are in a special case with interesting properties.

Let us now discuss the examples on lists considered at the beginning of this section. Recall the definitions of judgements $\mathsf{member}(x, l)$ and $\mathsf{allPos}(l)$:

$$\frac{}{\mathsf{member}(x, x{:}l)} \qquad \frac{\mathsf{member}(x, l)}{\mathsf{member}(x, y{:}l)} \qquad \frac{}{\mathsf{allPos}(\varepsilon)} \qquad \frac{\mathsf{allPos}(l)}{\mathsf{allPos}(x{:}l)}\ x > 0$$

where l ranges over finite lists of integers and x, y on integers. We interpret both inference systems inductively. The following are valid proof trees for some judgements:

$$\frac{}{\mathsf{member}(1, 1{:}2{:}1{:}\varepsilon)} \qquad \frac{\dfrac{}{\mathsf{member}(1, 1{:}\varepsilon)}}{\mathsf{member}(1, 2{:}1{:}\varepsilon)} \qquad \frac{\dfrac{\dfrac{}{\mathsf{allPos}(\varepsilon)}}{\mathsf{allPos}(1{:}\varepsilon)}}{\mathsf{allPos}(2{:}1{:}\varepsilon)}$$

Note that the same judgement can have different proof trees, as for $\mathsf{member}(1, 1{:}2{:}1{:}\varepsilon)$. This is due to the nature of meta-rules that are in some sense redundant: the second rule is applicable also in cases where the first suffices. In order to remove this redundancy, we can add a side condition to the second meta-rule, to make the two meta-rules mutually exclusive: the needed side condition is $x \neq y$. In this way, the second tree depicted above is not a proof tree since the first step is not justified by any rule.

Let us now assume that l ranges over both finite and infinite lists of integers. Now, what happens if we interpret both inference systems inductively? For $\mathsf{member}(x, l)$ we exactly derive all expected judgements, since it suffices to inspect finitely many elements of the list to check that x occurs in l. For $\mathsf{allPos}(l)$, instead, we cannot deal with infinite lists using finite proof trees. Intuitively, this is due to the fact that, to carry our a valid derivation, we need to inspect all the elements of the list and, if these are infinitely many, we cannot do it by a finite proof tree.

Therefore, to properly deal with infinite lists, in this case we need non-well-founded derivations, like the following one for the infinite list repeating

1 and 2,

$$\dfrac{\dfrac{\dfrac{\vdots}{\mathsf{allPos}(1{:}2{:}\ldots)}}{\mathsf{allPos}(2{:}1{:}2{:}\ldots)}}{\mathsf{allPos}(1{:}2{:}1{:}2{:}\ldots)}$$

which is constructed by applying infinitely many times the second (meta-)rule. Indeed, the coinductive interpretation is the correct one for the judgement $\mathsf{allPos}(l)$.

Now, what happens if we interpret the definition of $\mathsf{member}(x, l)$ coinductively? In this case we get a wrong semantics, because we can construct infinite proof trees for incorrect judgements, like the following one:

$$\dfrac{\dfrac{\dfrac{\vdots}{\mathsf{member}(0, 1{:}2{:}\ldots)}}{\mathsf{member}(0, 2{:}1{:}2{:}\ldots)}}{\mathsf{member}(0, 1{:}2{:}1{:}2{:}\ldots)}$$

This derivation is an infinite non-well-founded proof tree, since each step is correctly justified by a rule, but it proves a judgement that should not hold.

Let us conclude this section by showing an example dealing with another important non-well-founded structure: graphs. This is another case where coinduction is needed in order to correctly define judgements. We represent graphs by the adjacency function $G : V \to \wp(V)$, where V is the finite set of nodes, that is, for each $v \in V$, $G(v)$ is the set of nodes adjacent to v. We define the judgement $\mathsf{dist}_G(v, u, \delta)$, with $\delta \in \mathbb{N} + \{\infty\}$, which should hold iff δ is the distance from v to u, that is, the least length of a path from v to u, or, in other words, the least number of edges we have to traverse to go from v to u. The judgement is defined by the following (meta-)rules, where we assume $\min \emptyset = \infty$:

$$(\textsc{empty})\ \frac{}{\mathsf{dist}_G(v, v, 0)}$$

$$(\textsc{adj})\ \frac{\mathsf{dist}_G(v_1, u, \delta_1) \quad \ldots \quad \mathsf{dist}_G(v_n, u, \delta_n) \qquad v \neq u}{\mathsf{dist}_G(v, u, 1 + \min\{\delta_1, \ldots, \delta_n\}) \qquad G(v) = \{v_1, \ldots, v_n\}}$$

Since the definition follows the structure of the graph, the inductive interpretation is not enough: it can only deal with acyclic graphs, because, in presence of cycles, we have to deal with possibly infinite paths (*e.g.*, a finite path followed by a cycle), hence we cannot reach a base case (an axiom) in finitely many steps. Therefore, the above rules have to be interpreted coinductively, allowing non-well-founded derivations.

Consider the graph in Figure 2.2: we need infinite proofs to derive judgements like $\mathsf{dist}_G(a, c, 2)$ or $\mathsf{dist}_G(b, c, 1)$, since both a and b are part of a cycle. Note also that $\mathsf{dist}_G(c, v, \infty)$ is the only derivable judgement for all $v \in \{a, b, d\}$, since there are no outgoing edges from c, hence instances of (ADJ) have no premises and so $1 + \min\{\delta_1, \ldots, \delta_n\} = 1 + \min \emptyset = \infty$. Finally let us consider

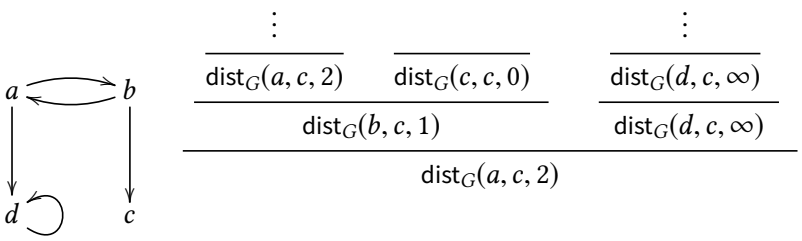


FIGURE 2.2 On the left side a concrete graph G with nodes $\{a, b, c, d\}$, and on the right side the non-well-founded derivation of the judgement $\mathsf{dist}_G(a, c, 2)$.

judgements of shape $\mathsf{dist}_G(d, c, \delta)$. A derivation scheme for these judgements is the following:

$$\dfrac{\dfrac{\dfrac{\vdots}{\mathsf{dist}_G(d,c,\delta-2)}}{\mathsf{dist}_G(d,c,\delta-1)}}{\mathsf{dist}_G(d,c,\delta)}$$

Now, which value of δ makes the proof correct? Surely for $\delta = \infty$ the proof is valid, because it becomes cyclic. Actually there is no other possible value, because, going up in the proof tree, δ should indefinitely decrease, and this is not possible since δ is a natural number and so it cannot go below zero. Therefore, as expected, $\mathsf{dist}_G(d, c, \infty)$ is the only derivable judgement, meaning that we cannot reach c starting from d.

2.2 Fixed points in complete lattices

In this section we recall basic definitions and results about complete lattices, a key notion for next sections and chapters. We refer to (Davey and Priestley, 2002) for more details.

A *partially ordered set*, *poset* for short, is a pair $\langle P, \sqsubseteq \rangle$, where P is a set and $\sqsubseteq$ is a partial order, that is, a reflexive, antisymmetric and transitive relation on P. Let $\langle P, \sqsubseteq \rangle$ be a poset. A *top* element of P is an element $z \in P$ such that $x \sqsubseteq z$, for all $x \in P$, and, dually, a *bottom* element is an element $z \in P$ such that $z \sqsubseteq x$, for all $x \in P$. Both top and bottom elements are unique and denoted by $\top$ and $\bot$, respectively. Furthermore, given a subset $A \subseteq P$, a *least upper bound* (a.k.a. *supremum* or *join*) of A is an element $z \in P$ such that $x \sqsubseteq z$, for all $x \in A$, and, for all $z' \in P$ such that $x \sqsubseteq z'$ for all $x \in A$, $z \sqsubseteq z'$. The least upper bound of A is unique and we denote it by $\bigsqcup A$. The *greatest lower bound* (a.k.a. *infimum* or *meet*), denoted by $\bigsqcap A$, is defined dually as an element $z \in P$ such that $z \sqsubseteq x$, for all $x \in A$, and, for all $z' \in P$ such that $z' \sqsubseteq x$ for all $x \in A$, $z' \sqsubseteq z$. We can now define complete lattices:

DEFINITION 2.7 : A poset $\langle L, \sqsubseteq \rangle$ is a *complete lattice* if all subsets $A \subseteq L$ have a least upper bound $\bigsqcup A$.

The paradigmatic example of complete lattice is the power-set lattice of a set X, $\langle \wp(X), \subseteq \rangle$, where the carrier set is the set of all subsets of X, the order is set inclusion and the least upper bound is given by union.

From the definition, we immediately get that a complete lattice $\langle L, \sqsubseteq \rangle$ has a top element $\top = \bigsqcup L$, a bottom element $\bot = \bigsqcup \emptyset$ and all subsets $A \subseteq L$ have a greatest lower bound $\bigsqcap A = \bigsqcup \{z \in L \mid z \sqsubseteq x, \text{ for all } x \in A\}$. In the following, we will use infix notation for binary versions of join and meet operations. In the power-set lattice $\langle \wp(X), \subseteq \rangle$ we have $\top = X$, $\bot = \emptyset$ and, for all $A \subseteq \wp(X)$ (A is a set of subsets of X), $\bigsqcap A = \bigcap A$.

We now introduce the class of functions between posets we will be mainly interested in: monotone functions, that is, functions preserving the order structure.

DEFINITION 2.8 : Let $\langle P, \sqsubseteq_P \rangle$ and $\langle Q, \sqsubseteq_Q \rangle$ be posets. A function $F : P \to Q$ is *monotone* if, for all $x, y \in P$, if $x \sqsubseteq_P y$ then $F(x) \sqsubseteq_Q F(y)$.

A function $F : P \to P$ on a poset $\langle P, \sqsubseteq \rangle$ identifies three subsets of elements of P, which play a central role: let $x \in P$ be an element, we say that

- x is a *pre-fixed point* of F if $F(x) \sqsubseteq x$,
- x is a *post-fixed point* of F if $x \sqsubseteq F(x)$,
- x is a *fixed point* of F if $x = F(x)$.

We will denote by $\mathsf{pre}(F)$, $\mathsf{post}(F)$ and $\mathsf{fix}(F)$ the sets of pre-fixed points, post-fixed points and fixed points of F, respectively.

If F is a monotone function on a complete lattice $\langle L, \sqsubseteq \rangle$, $\mathsf{pre}(F)$ and $\mathsf{post}(F)$ have an important property: they are closed under arbitrary meets and arbitrary joins in L, respectively.

PROPOSITION 2.9 : Let $\langle L, \sqsubseteq \rangle$ be a complete lattice and $F : L \to L$ a monotone function:

1. if $A \subseteq \mathsf{pre}(F)$ then $\bigsqcap A \in \mathsf{pre}(F)$, and
2. if $A \subseteq \mathsf{post}(F)$ then $\bigsqcup A \in \mathsf{post}(F)$.

Proof: We prove the second point, the other one follows by duality. Assume $A \subseteq \mathsf{post}(F)$, since for all $x \in A$ we have $x \sqsubseteq \bigsqcup A$, by monotonicity we get $F(x) \sqsubseteq F(\bigsqcup A)$. Then, since $x \in A \subseteq \mathsf{post}(F)$, we have $x \sqsubseteq F(x)$, thus we get $x \sqsubseteq F(\bigsqcup A)$ and this implies $\bigsqcup A \sqsubseteq F(\bigsqcup A)$, as needed. □

Monotone functions over a complete lattice enjoys a very importante property: they have a least and a greatest fixed point. This result is known as the *Knaster-Tarski fixed point theorem* (Tarski, 1955). We report the statement below:

THEOREM 2.10 (Knaster-Tarski): Let $\langle L, \sqsubseteq \rangle$ be a complete lattice and let $F : L \to L$ be a monotone function. Then, F has both a least and a greatest fixed point, denoted by μF and νF, respectively, and defined by

$$\mu F = \bigsqcap \mathsf{pre}(F) \qquad \nu F = \bigsqcup \mathsf{post}(F)$$

Proof: We prove the first point, the other follows by duality. Set $z = \bigsqcap \mathsf{pre}(F)$, then, by Proposition 2.9 (1), we get $F(z) \sqsubseteq z$. Then, by monotonicity, we have $F(F(z)) \sqsubseteq F(z)$, that is, $F(z) \in \mathsf{pre}(F)$, hence $z = \bigsqcap \mathsf{pre}(F) \sqsubseteq F(z)$. This shows that z is a fixed point of F. To prove it is the least one, just note that any fixed point w is also pre-fixed, thus $z \sqsubseteq w$, as needed. □

In other words, the least pre-fixed point and the greatest post-fixed point are fixed points, hence they are the least and the greatest one. As an immediate consequence of this theorem, we get the following properties of μF and νF: let $x \in L$, then

(μP) if $F(x) \sqsubseteq x$ then $\mu F \sqsubseteq x$, and

(νP) if $x \sqsubseteq F(x)$ then $x \sqsubseteq \nu F$.

We conclude this section by discussing an alternative, iterative, characterisation of μF and νF, under additional assumptions on F. First of all, let us introduce some basic definitions. A sequence $(x_i)_{i \in \mathbb{N}}$, with $x_i \in L$ for all $i \in \mathbb{N}$, is an *increasing ω-chain* if, for all $i \in \mathbb{N}$, $x_i \sqsubseteq x_{i+1}$, and it is a *decreasing ω-chain* if, for all $i \in \mathbb{N}$, $x_{i+1} \sqsubseteq x_i$. We are now interested in *continuous* functions in the following sense:

DEFINITION 2.11 : Let $F : L \to L$ be a function. We say that

- F is *upward ω-continuous* if, for any increasing ω-chain $(x_i)_{i \in \mathbb{N}}$, we have $F(\bigsqcup_{i \in \mathbb{N}} x_i) = \bigsqcup_{i \in \mathbb{N}} F(x_i)$, and
- F is *downward ω-continuous* if, for any decreasing ω-chain $(x_i)_{i \in \mathbb{N}}$, we have $F(\bigsqcap_{i \in \mathbb{N}} x_i) = \bigsqcap_{i \in \mathbb{N}} F(x_i)$.

Sometimes, upward ω-continuous functions are simply called continuous and downward continuous functions are called cocontinuous, *e.g.*, by Sangiorgi (2011). An important property to note is the following:

PROPOSITION 2.12 : If $F : L \to L$ is upward or downward ω-continuous, then it is monotone.

Proof: We prove the thesis assuming F to be upward ω-continuous, the other case is analogous. Let $x, y \in L$ be such that $x \sqsubseteq y$ and define the sequence $(x_i)_{i \in \mathbb{N}}$ as follows: $x_0 = x$ and $x_i = y$ for all $i > 0$. Clearly $(x_i)_{i \in \mathbb{N}}$ is an increasing ω-chain and $\bigsqcup_{i \in \mathbb{N}} x_i = y$. Then, we get $F(x) = F(x_0) \sqsubseteq \bigsqcup_{i \in \mathbb{N}} F(x_i) = F(\bigsqcup_{i \in \mathbb{N}}) = F(y)$, as needed. □

Therefore, by Theorem 2.10, we get that ω-continuous functions over a complete lattice admit least and greatest fixed points, as they are also monotone. However, for ω-continuous functions, we can provide an iterative characterisation of least and greatest fixed points. Given a function $F : L \to L$, for any

natural number $n \in \mathbb{N}$, let us define the function $F^n : L \to L$, the *n-iteration of* F, by induction as follows:

$$F^0(x) = x$$
$$F^{n+1}(x) = F(F^n(x))$$

for all $x \in L$. Then, given an element $x \in L$, the sequence $(F^n(x))_{n\in\mathbb{N}}$ is the sequence of *iterates of* F *on* x, which has the following properties:

PROPOSITION 2.13 : Let $F : L \to L$ be a monotone function and $x \in L$. The following hold:

- if $x \sqsubseteq F(x)$ then $(F^n(x))_{n\in\mathbb{N}}$ is an increasing ω-chain, and
- if $F(x) \sqsubseteq x$ then $(F^n(x))_{n\in\mathbb{N}}$ is a decreasing ω-chain.

Proof: We prove only the first point, the other follows by duality. Assume $x \sqsubseteq F(x)$, then, by induction on n, we prove that $F^n(x) \sqsubseteq F^{n+1}(x)$. If $n = 0$ the thesis holds by hypothesis; otherwise, by induction hypothesis we know that $F^n(x) \sqsubseteq F^{n+1}(x)$, hence, as F is monotone, we get $F^{n+1}(x) = F(F^n(x)) \sqsubseteq F(F^{n+1}(x)) = F^{n+2}(x)$, as needed. □

Finally, under continuity assumptions, we get the iterative characterisation of least and greatest fixed points. This result is known as Kleene's theorem.

THEOREM 2.14 (Kleene): Let $F : L \to L$ be a function. The following hold:

- if F is upward ω-continuous then $\mu F = \bigsqcup_{n\in\mathbb{N}} F^n(\bot)$;
- if F is downward ω-continuous then $\nu F = \bigsqcap_{n\in\mathbb{N}} F^n(\top)$.

Proof: We prove the first point, the other one follows by duality. Assume F to be upward ω-continuous, hence, by Proposition 2.12, it is monotone as well. Set $z = \bigsqcup_{n\in\mathbb{N}} F^n(\bot)$. Since $\bot \sqsubseteq F(\bot)$, by Proposition 2.13, the sequence $(F^n(\bot))_{n\in\mathbb{N}}$ is increasing, then, by continuity, we get $F(z) = \bigsqcup_{n\in\mathbb{N}} F(F^n(\bot)) = \bigsqcup_{n\in\mathbb{N}} F^{n+1}(\bot) = \bot \sqcup \bigsqcup_{n\in\mathbb{N}} F^{n+1}(\bot) = z$, that is, z is a fixed point of F. To prove z is the least fixed point, by Theorem 2.10, we just have to prove that, for any $w \in \mathsf{pre}(F)$, we have $z \sqsubseteq w$. By induction on n, we show that $F^n(\bot) \sqsubseteq w$, for all $n \in \mathbb{N}$, and this will imply the thesis. If $n = 0$, we trivially get $F^0(\bot) = \bot \sqsubseteq w$. Then, by induction hypothesis, we have $F^n(\bot) \sqsubseteq w$, hence, as F is monotone and w is pre-fixed, we get $F^{n+1}(\bot) = F(F^n(\bot)) \sqsubseteq F(w) \sqsubseteq w$, as needed. □

2.3 Fixed point semantics

In this section we will describe the inductive and coinductive interpretations of an inference system in terms of fixed points of monotone functions associated with it. As a result, we will get an equivalent purely model-theoretic definition of such interpretations, that is, a definition independent from the notion of proof tree.

Assume an inference system $\mathcal{I}$ on the universe $\mathcal{U}$. We can associate with $\mathcal{I}$ a function $F_{\mathcal{I}} : \wp(\mathcal{U}) \to \wp(\mathcal{U})$, called the *inference operator* and defined as follows: :

$$F_{\mathcal{I}}(X) = \{j \in \mathcal{U} \mid \langle Pr, j\rangle \in \mathcal{I}, \text{ for some } Pr \subseteq X\}$$

This function maps a set of judgements $X \subseteq \mathcal{U}$ to the set of judgements that can be derived from X by applying a rule in $\mathcal{I}$. This is the reason why it is called inference operator, as it models the action of deriving new judgements starting from given ones.

It is easy to see that properties of sets of judgements introduced in Definition 2.2 can be rephrased using the inference operator. Indeed, if $X \subseteq \mathcal{U}$ is a set of judgements, then X is $\mathcal{I}$-closed iff it is a pre-fixed point of $F_{\mathcal{I}}$, namely, $F_{\mathcal{I}}(X) \subseteq X$; X is $\mathcal{I}$-consistent iff it is a post-fixed point of $F_{\mathcal{I}}$, namely, $X \subseteq F_{\mathcal{I}}(X)$, and X is an $\mathcal{I}$-interpretation iff it is a fixed point of $F_{\mathcal{I}}$, namely, $F_{\mathcal{I}}(X) = X$. Hence, to construct interpretations of $\mathcal{I}$, we just have to construct fixed points of $F_{\mathcal{I}}$.

Since $\langle \wp(\mathcal{U}), \subseteq\rangle$ is a complete lattice, the key property, that allows us to construct fixed points of $F_{\mathcal{I}}$, is the following:

PROPOSITION 2.15 : The function $F_{\mathcal{I}} : \wp(\mathcal{U}) \to \wp(\mathcal{U})$ is monotone with respect to set inclusion.

Indeed, by the Knaster-Tarski theorem (Theorem 2.10), we know $F_{\mathcal{I}}$ has least and greatest fixed points, $\mu F_{\mathcal{I}}$ and $\nu F_{\mathcal{I}}$, and they coincide with the least pre-fixed point and the greatest post-fixed point, respectively. In other words, $\mu F_{\mathcal{I}}$ is the least $\mathcal{I}$-closed set and $\nu F_{\mathcal{I}}$ is the greatest $\mathcal{I}$-consistent set. In the following we will show that these two fixed points coincide with the inductive and the coinductive interpretation as defined in Definition 2.6, thus obtaining a purely model-theoretic definition of these two interpretations of $\mathcal{I}$.

Rather than giving ad-hoc proofs, we present a general framework where to express in a uniform and systematic way the equivalence between proof-theoretic and model-theoretic semantics, and then state and prove such equivalence for the inductive and the coinductive interpretations.

Let $\mathcal{T}_{\mathcal{U}}$ be the set of all trees on the universe $\mathcal{U}$ and let $\mathsf{r} : \mathcal{T}_{\mathcal{U}} \to \mathcal{U}$ be the function that maps a tree to its root. Then, the direct image and the inverse image along r are $\mathsf{r}_! : \wp(\mathcal{T}_{\mathcal{U}}) \to \wp(\mathcal{U})$, and $\mathsf{r}^* : \wp(\mathcal{U}) \to \wp(\mathcal{T}_{\mathcal{U}})$, respectively. The fundamental fact is that the functions $\mathsf{r}_!$ and r^* are related by an adjunction $\mathsf{r}_! \dashv \mathsf{r}^*$, that is, for all $X \subseteq \mathcal{T}_{\mathcal{U}}$ and $Y \subseteq \mathcal{U}$, $\mathsf{r}_!(X) \subseteq Y$ iff $X \subseteq \mathsf{r}^*(Y)$. In other words, $\mathsf{r}_!$ behaves as an abstraction function (Cousot and Cousot, 1977), as it forgets about trees. Intuitively, when acting on proof trees, $\mathsf{r}_!$ maps a set of proofs to the set of judgements they prove.

From the inference system $\mathcal{I}$, we can define an inference operator on sets of trees, called the *tree inference operator*, defined as follows:

$$T_{\mathcal{I}}(Y) = \{\tau \in \mathcal{T}_{\mathcal{U}} \mid \mathsf{dst}(\tau) \subseteq Y \text{ and } \langle \mathsf{r}_!(\mathsf{dst}(\tau)), \mathsf{r}(\tau)\rangle \in \mathcal{I}\}$$

This function behaves very much like $F_{\mathcal{I}}$, indeed it maps a set of trees $Y \subseteq \mathcal{T}_{\mathcal{U}}$ to the set of trees that can be built starting from those in Y by applying a rule

in $\mathcal{I}$. Basically, $F_{\mathcal{I}}$ can be regarded as an abstract version of $T_{\mathcal{I}}$, which is more concrete because it keeps track of the trees used to derive the premises of the applied rule. The next proposition makes this observation formal, by relying on the adjunction $\mathsf{r}_! \dashv \mathsf{r}^*$, which, as already mentioned, models the abstraction from trees to judgements.

PROPOSITION 2.16 : $\mathsf{r}_! \circ T_{\mathcal{I}} = F_{\mathcal{I}} \circ \mathsf{r}_!$ and $T_{\mathcal{I}} \circ \mathsf{r}^* \subseteq \mathsf{r}^* \circ F_{\mathcal{I}}$.

Proof: Towards a proof of $\mathsf{r}_! \circ T_{\mathcal{I}} = F_{\mathcal{I}} \circ \mathsf{r}_!$, note that, for all $Y \subseteq \mathcal{T}_{\mathcal{U}}$, if $\tau \in \mathsf{r}_!(T_{\mathcal{I}}(Y))$, then $\mathsf{r}_!(\mathsf{dst}(\tau)) \subseteq Y$ and $\langle \mathsf{r}_!(\mathsf{dst}(\tau)), \mathsf{r}(\tau)\rangle \in \mathcal{I}$, hence $\mathsf{r}(\tau) \in F_{\mathcal{I}}(\mathsf{r}_!(Y))$, and this proves $\mathsf{r}_! \circ T_{\mathcal{I}} \subseteq F_{\mathcal{I}} \circ \mathsf{r}_!$. To get the other inclusion, if $c \in F_{\mathcal{I}}(\mathsf{r}_!(Y))$, then $\langle Pr, c\rangle \in \mathcal{I}$, for some $Pr \subseteq \mathsf{r}_!(Y)$, hence, for all $j \in Pr$, there is $\tau_j \in Y$ such that $\mathsf{r}(\tau_j) = j$. We choose a tree τ_j for each $j \in Pr$ and denote by Z the set of such trees. Then, $\tau = \dfrac{Z}{c}$ is a tree and $\mathsf{dst}(\tau) = Z \subseteq Y$, hence $\tau \in T_{\mathcal{I}}(Y)$ and so $c = \mathsf{r}(\tau) \in \mathsf{r}_!(T_{\mathcal{I}}(Y))$, as needed.

Towards a proof of $T_{\mathcal{I}} \circ \mathsf{r}^* \subseteq \mathsf{r}^* \circ F_{\mathcal{I}}$, note that, for all $X \subseteq \mathcal{U}$, if $\tau \in T_{\mathcal{I}}(\mathsf{r}^*(X))$, then $\mathsf{dst}(\tau) \subseteq \mathsf{r}^*(X)$ and $\langle \mathsf{r}_!(\mathsf{dst}(\tau)), \mathsf{r}(\tau)\rangle \in \mathcal{I}$, hence, $\mathsf{r}_!(\mathsf{dst}(\tau)) \subseteq X$ and so $\mathsf{r}(\tau) \in F_{\mathcal{I}}(X)$ and this implies $\tau \in \mathsf{r}^*(F_{\mathcal{I}}(X))$, as needed. □

From the adjunction $\mathsf{r}_! \dashv \mathsf{r}^*$ and the above proposition we immediately get the following corollary:

COROLLARY 2.17 : Let $X \subseteq \mathcal{T}_{\mathcal{U}}$ and $Y \subseteq \mathcal{U}$, then

- if $X \subseteq T_{\mathcal{I}}(X)$ then $\mathsf{r}_!(X) \subseteq F_{\mathcal{I}}(\mathsf{r}_!(X))$,
- if $T_{\mathcal{I}}(X) \subseteq X$ then $F_{\mathcal{I}}(\mathsf{r}_!(X)) \subseteq \mathsf{r}_!(X)$, and
- if $F_{\mathcal{I}}(Y) \subseteq Y$ then $T_{\mathcal{I}}(\mathsf{r}^*(Y)) \subseteq \mathsf{r}^*(Y)$.

In other words, the direct image $\mathsf{r}_!$ maps pre-fixed, post-fixed and fixed points of $T_{\mathcal{I}}$ to pre-fixed, post-fixed and fixed points of $F_{\mathcal{I}}$, and the function r^* maps pre-fixed points of $F_{\mathcal{I}}$ to pre-fixed points of $T_{\mathcal{I}}$.

Fixed points of $T_{\mathcal{I}}$ will play an important role to characterise the proof-theoretic semantics of $\mathcal{I}$. In particular, as shown by the following lemma, post-fixed points of $T_{\mathcal{I}}$ characterise proof trees in $\mathcal{I}$.

LEMMA 2.18 : The following hold:

1. If $X \subseteq T_{\mathcal{I}}(X)$, then all $\tau \in X$ are proof trees in $\mathcal{I}$.
2. If τ is a proof tree in $\mathcal{I}$, then $\mathsf{SubTr}(\tau) \subseteq T_{\mathcal{I}}(\mathsf{SubTr}(\tau))$.

Proof:

1. Let $\tau \in X$, we have to show that, for all $\alpha \in \mathsf{N}(\tau)$, $\langle \mathsf{r}_!(\mathsf{chl}_\tau(\alpha)), \tau(\alpha)\rangle \in \mathcal{I}$. We prove by induction on α that, for all $\alpha \in \mathsf{N}(\tau)$, $\tau_{|_\alpha} \in X$. If $\alpha = \varepsilon$, then $\tau_{|_\varepsilon} = \tau \in X$ by hypothesis. If $\alpha = \beta j$, then $\tau_{|_{\beta j}} = (\tau_{|_\beta})_{|_j} \in \mathsf{dst}(\tau_{|_\beta})$ and, by induction hypothesis, we have $\tau_{|_\beta} \in X$. Since $X \subseteq T_{\mathcal{I}}(X)$, we have $\mathsf{dst}(\tau_{|_\beta}) \subseteq X$, as needed.

To conclude, note that, for all $\alpha \in \mathsf{N}(\tau)$, $\mathsf{chl}_\tau(\alpha) = \mathsf{dst}(\tau_{|_\alpha})$ and $\tau(\alpha) = \mathsf{r}(\tau_{|_\alpha})$, hence, since $\tau_{|_\alpha} \in X \subseteq T_{\mathcal{I}}(X)$, as we have just proved, we have $\langle \mathsf{r}_!(\mathsf{dst}(\tau_{|_\alpha})), \mathsf{r}(\tau_{|_\alpha})\rangle \in \mathcal{I}$, as needed.

2. Let $\tau' \in \mathsf{SubTr}(\tau)$, then $\tau' = \tau_{|_\alpha}$ for some $\alpha \in \mathsf{N}(\tau)$. Since τ is a proof tree in $\mathcal{I}$, we have $\langle \mathsf{r}_!(\mathsf{chl}_\tau(\alpha)), \tau(\alpha)\rangle \in \mathcal{I}$, then, since $\tau(\alpha) = \mathsf{r}(\tau_{|_\alpha})$ and $\mathsf{chl}_\tau(\alpha) = \mathsf{dst}(\tau_{|_\alpha}) \subseteq \mathsf{SubTr}(\tau)$, we have $\tau_{|_\alpha} \in T_{\mathcal{I}}(\mathsf{SubTr}(\tau))$, as needed.

□

Note that, as a consequence, any fixed point of $T_{\mathcal{I}}$ contains only proof trees.

Since $\langle \wp(\mathcal{T}_{\mathcal{U}}), \subseteq\rangle$ is a complete lattice, to ensure the existence of the least and the greatest fixed points of $T_{\mathcal{I}}$, we just have to prove it is monotone:

PROPOSITION 2.19 : The function $T_{\mathcal{I}} : \wp(\mathcal{T}_{\mathcal{U}}) \to \wp(\mathcal{T}_{\mathcal{U}})$ is monotone with respect to set inclusion.

Therefore, the least and the greatest fixed points of $T_{\mathcal{I}}$ exist and they play a crucial role in the proof-theoretic definition of the inductive and the coinductive interpretations of $\mathcal{I}$, as the following results show.

LEMMA 2.20 : $\mu T_{\mathcal{I}}$ is the set of well-founded proof trees in $\mathcal{I}$.

Proof: Let $W \subseteq \mathcal{T}_{\mathcal{U}}$ be the set of well-founded proof trees in $\mathcal{I}$. Recall that $\tau \prec \tau'$ iff $\tau \in \mathsf{dst}(\tau')$, and this relation is well-founded on W, because all trees in W are well-founded. Therefore, we can prove $W \subseteq \mu T_{\mathcal{I}}$ by well-founded induction on $\prec$. Let $\tau \in W$ and assume that $\tau' \in \mu T_{\mathcal{I}}$, for all $\tau' \in \mathsf{dst}(\tau)$. Since τ is a proof tree (see Definition 2.5), we know that $\langle \mathsf{r}_!(\mathsf{dst}(\tau)), \mathsf{r}(\tau)\rangle \in \mathcal{I}$ and, by induction hypothesis, we know that $\mathsf{dst}(\tau) \subseteq \mu T_{\mathcal{I}}$, hence, by definition of $T_{\mathcal{I}}$, we get $\tau \in T_{\mathcal{I}}(\mu T_{\mathcal{I}}) = \mu T_{\mathcal{I}}$, as needed.

On the other hand, to prove $\mu T_{\mathcal{I}} \subseteq W$, by Theorem 2.10 we just have to prove that $T_{\mathcal{I}}(W) \subseteq W$. Let $\tau \in T_{\mathcal{I}}(W)$, then, by definition of $T_{\mathcal{I}}$, we have $\langle \mathsf{r}_!(\mathsf{dst}(\tau)), \mathsf{r}(\tau)\rangle \in \mathcal{I}$ and $\mathsf{dst}(\tau) \subseteq W$, hence, in particular, all direct subtrees of τ are well-founded proof trees. Therefore, τ is a well-founded proof tree as well, thus $\tau \in W$, as needed. □

LEMMA 2.21 : $\nu T_{\mathcal{I}}$ is the set of all (well-founded or not) proof trees in $\mathcal{I}$.

Proof: Let $Z \subseteq \mathcal{T}_{\mathcal{U}}$ be the set of all proof trees in $\mathcal{I}$. We first prove the inclusion $\nu T_{\mathcal{I}} \subseteq Z$. Since $\nu T_{\mathcal{I}}$ is a post-fixed point of $T_{\mathcal{I}}$, by Lemma 2.18 (1), we get the inclusion. To prove the other inclusion $Z \subseteq \nu T_{\mathcal{I}}$, just note that, if $\tau \in Z$, then τ is a proof tree, hence, by Lemma 2.18 (2), we get $\mathsf{SubTr}(\tau)$ is a post-fixed point of $T_{\mathcal{I}}$. Therefore, by Theorem 2.10, we get $\tau \in \mathsf{SubTr}(\tau) \subseteq \nu T_{\mathcal{I}}$, as needed. □

Building on these lemmas, we can rephrase the proof-theoretic definition of the inductive and the coinductive interpretations (Definition 2.6) as follows:

COROLLARY 2.22 : $\mu[\![\mathcal{I}]\!] = \mathsf{r}_!(\mu T_{\mathcal{I}})$ and $\nu[\![\mathcal{I}]\!] = \mathsf{r}_!(\nu T_{\mathcal{I}})$.

We can now prove the two main results of this section, showing that the inductive and the coinductive interpretations are fixed points of $F_{\mathcal{I}}$. More precisely, $\mu[\![\mathcal{I}]\!]$ coincides with the least fixed point of $F_{\mathcal{I}}$, while $\nu[\![\mathcal{I}]\!]$ coincides with the greatest fixed point of $F_{\mathcal{I}}$.

We start from the inductive case. The result in this case is an immediate consequence of the adjoint situation $\mathsf{r}_! \dashv \mathsf{r}^*$, by the so called *fusion rule* (Davey and Priestley, 2002). We give an explicit proof for sake of completeness.

THEOREM 2.23 : $\mu[\![\mathcal{I}]\!] = \mu F_{\mathcal{I}}$.

Proof: By Corollary 2.22, we have $\mu[\![\mathcal{I}]\!] = \mathsf{r}_!(\mu T_{\mathcal{I}})$. Since $T_{\mathcal{I}}(\mu T_{\mathcal{I}}) \subseteq \mu T_{\mathcal{I}}$, by Corollary 2.17 we get $F_{\mathcal{I}}(\mathsf{r}_!(\mu T_{\mathcal{I}})) \subseteq \mathsf{r}_!(\mu T_{\mathcal{I}})$, hence, by Theorem 2.10, we get $\mu F_{\mathcal{I}} \subseteq \mathsf{r}_!(\mu T_{\mathcal{I}})$. To prove the other inclusion, since $F_{\mathcal{I}}(\mu F_{\mathcal{I}}) \subseteq \mu F_{\mathcal{I}}$, by Corollary 2.17 we get $T_{\mathcal{I}}(\mathsf{r}^*(\mu F_{\mathcal{I}})) \subseteq \mathsf{r}^*(\mu F_{\mathcal{I}})$, hence, by Theorem 2.10, we get $\mu T_{\mathcal{I}} \subseteq \mathsf{r}^*(\mu F_{\mathcal{I}})$. Then, by the adjunction $\mathsf{r}_! \dashv \mathsf{r}^*$, we get $\mathsf{r}_!(\mu T_{\mathcal{I}}) \subseteq \mu F_{\mathcal{I}}$, as needed. □

In the coinductive case, the proof is not immediate and it relies on Theorem 2.4, as detailed below.

THEOREM 2.24 : $\nu[\![\mathcal{I}]\!] = \nu F_{\mathcal{I}}$.

Proof: By Corollary 2.22, we have $\nu[\![\mathcal{I}]\!] = \mathsf{r}_!(\nu T_{\mathcal{I}})$. Since $\nu T_{\mathcal{I}} \subseteq T_{\mathcal{I}}(\nu T_{\mathcal{I}})$, by Corollary 2.17 we get $\mathsf{r}_!(\nu T_{\mathcal{I}}) \subseteq F_{\mathcal{I}}(\mathsf{r}_!(\nu T_{\mathcal{I}}))$, hence, by Theorem 2.10, we get $\mathsf{r}_!(\nu T_{\mathcal{I}}) \subseteq \nu F_{\mathcal{I}}$. To prove the other inclusion, we just have to show that, given a set $X \subseteq \mathcal{U}$ such that $X \subseteq F_{\mathcal{I}}(X)$, each judgement $j \in X$ is the root of a proof tree. Since $X \subseteq F_{\mathcal{I}}(X)$, X is consistent (Definition 2.2), that is, for each $j \in X$, there is $Pr_j \subseteq X$ such that $\langle Pr_j, j\rangle \in \mathcal{I}$. Hence, applying Theorem 2.4, where g maps j to Pr_j and v is the restriction of the identity on $\mathcal{U}$ to X, we get an injective function $p : X \to \mathcal{T}_{\mathcal{U}}$ which makes the diagram in Theorem 2.4 commute. We have still to prove that $p(j)$ is a proof tree. To this end, by Lemma 2.18 (1), we just have to show that the set $p_!(X) = \{p(j) \mid j \in X\}$ is a post-fixed point of $T_{\mathcal{I}}$. By commutativity of the diagram, we have $\mathsf{dst}(p(j)) = p_!(g(j)) \subseteq p_!(X)$, $\mathsf{r}(p(j)) = j$ and $\mathsf{r}_!(\mathsf{dst}(p(j))) = Pr_j$, hence, as $\langle Pr_j, j\rangle \in \mathcal{I}$, we get $p(j) \in T_{\mathcal{I}}(p_!(X))$, as needed. □

2.4 Reasoning by (co)induction

In this section we discuss proof principles associated with the inductive and the coinductive interpretations of inference systems. Such proof principles are an immediate consequence of the fixed point characterisation of these interpretations provided in Theorem 2.23 and Theorem 2.24.

Let $\mathcal{I}$ be an inference system on the universe $\mathcal{U}$. We are typically interested in comparing the chosen interpretation of $\mathcal{I}$, say $[\![\mathcal{I}]\!]$, to a given set of judgements $\mathcal{S} \subseteq \mathcal{U}$ (*specification*). In particular, we focus on two properties:

SOUNDNESS all derivable judgements belong to $\mathcal{S}$, that is, $[\![\mathcal{I}]\!] \subseteq \mathcal{S}$,

COMPLETENESS all judgements in $\mathcal{S}$ are derivable, that is, $\mathcal{S} \subseteq [\![\mathcal{I}]\!]$.

In other words, if we look at $\mathcal{S}$ as a property of judgements, soundness tells us that all derivable judgements satisfy $\mathcal{S}$, while completeness tells us that all judgements satisfying $\mathcal{S}$ are derivable.

The inductive interpretation $\mu[\![\mathcal{I}]\!]$ comes with a proof principle for proving soundness. Such principle is the *induction principle*, stated below:

PROPOSITION 2.25 (Induction principle): Let $\mathcal{S} \subseteq \mathcal{U}$ be a set of judgements. If $\mathcal{S}$ is $\mathcal{I}$-closed, then $\mu[\![\mathcal{I}]\!] \subseteq \mathcal{S}$.

Proof: By Theorem 2.23 and Theorem 2.10, we have $\mu[\![\mathcal{I}]\!] = \mu F_{\mathcal{I}} = \bigcap \mathsf{pre}(F_{\mathcal{I}})$. Since $\mathcal{S}$ is $\mathcal{I}$-closed iff $\mathcal{S} \in \mathsf{pre}(F_{\mathcal{I}})$, we get the thesis. □

Spelling out the above principle, to prove that the inductive interpretation is sound with respect to a specification $\mathcal{S} \subseteq \mathcal{U}$, we just have to prove that $\mathcal{S}$ is $\mathcal{I}$-closed, that is, by Definition 2.2, for each rule $\langle Pr, c\rangle \in \mathcal{I}$, if $Pr \subseteq \mathcal{S}$ then $c \in \mathcal{S}$ as well. In other words, we have to prove that $\mathcal{S}$ holds for c assuming it for all premises $j \in Pr$. These assumptions are called *induction hypotheses*.

EXAMPLE 2.26 : Let us illustrate such principle by an example: recall from Section 2.1 the inference system $\mathcal{I}^{\mathsf{mem}}$ defining the judgement $\mathsf{member}(x, l)$, which should hold when x belongs to the list l:

$$(\text{M-H})\ \frac{}{\mathsf{member}(x, x{:}l)} \qquad (\text{M-T})\ \frac{\mathsf{member}(x, l)}{\mathsf{member}(x, y{:}l)}$$

We want to prove the following soundness statement:

if $\mathcal{I}^{\mathsf{mem}} \vdash_{\mu} \mathsf{member}(x, l)$ then x belongs to l.

Equivalently, if we set $\mathsf{member}(x, l) \in \mathcal{S}^{\mathsf{mem}}$ iff x belongs to l, the soundness statement can be expressed by the inclusion $\mu[\![\mathcal{I}^{\mathsf{mem}}]\!] \subseteq \mathcal{S}^{\mathsf{mem}}$.

We prove it by induction on rules in $\mathcal{I}^{\mathsf{mem}}$. We have two types of rules, hence we distinguish two cases:

Case: (M-H) We have no assumptions, as the rule is an axiom, hence we have just to prove that x belongs to $x{:}l$. But this is trivial, as x is the first element of $x{:}l$.

Case: (M-T) We assume that x belongs to l and we have to prove that x belongs to $y{:}l$. Again, this is trivial as x belongs to the tail of $y{:}l$ by assumption.

Dually, the coinductive interpretation $\nu[\![\mathcal{I}]\!]$ comes with a proof principle for proving completeness. Such principle is the *coinduction principle*, stated below:

PROPOSITION 2.27 (Coinduction principle): Let $\mathcal{S} \subseteq \mathcal{U}$ be a set of judgements. If $\mathcal{S}$ is $\mathcal{I}$-consistent, then $\mathcal{S} \subseteq \nu[\![\mathcal{I}]\!]$.

Proof: By Theorem 2.24 and Theorem 2.10, we have $\nu[\![\mathcal{I}]\!] = \nu F_{\mathcal{I}} = \bigcup \mathsf{post}(F_{\mathcal{I}})$. Since $\mathcal{S}$ is $\mathcal{I}$-consistent iff $\mathcal{S} \in \mathsf{post}(F_{\mathcal{I}})$, we get the thesis. □

Spelling out the above principle, to prove that the coinductive interpretation is complete with respect to a specification $\mathcal{S} \subseteq \mathcal{U}$, we just have to prove that $\mathcal{S}$ is $\mathcal{I}$-consistent, that is, by Definition 2.2, for each $j \in \mathcal{S}$, there is a rule $\langle Pr, j\rangle \in \mathcal{I}$ such that $Pr \subseteq \mathcal{S}$.

EXAMPLE 2.28 : Let us illustrate such principle by an example: recall from Section 2.1 the inference system $\mathcal{I}^{>0}$ defining the judgement $\mathsf{allPos}(l)$, which should hold when l is positive, that is, it contains only strictly positive elements

$$\text{(A-E)}\ \frac{}{\mathsf{allPos}(\varepsilon)} \qquad \text{(A-T)}\ \frac{\mathsf{allPos}(l)}{\mathsf{allPos}(x{:}l)}\quad x > 0$$

We want to prove the following completeness statement:

if l is positive, then $\mathcal{I}^{>0} \vdash_{\nu} \mathsf{allPos}(l)$.

Equivalently, if we set $\mathsf{allPos}(l) \in \mathcal{S}^{>0}$ iff l is positive, the completeness statement can be expressed by the inclusion $\mathcal{S}^{>0} \subseteq \nu[\![\mathcal{I}]\!]$.

We prove it by coinduction on rules in $\mathcal{I}^{>0}$. Assume $\mathsf{allPos}(l) \in \mathcal{S}^{>0}$, then l is positive, and distinguish two cases on l.

Case: $l = \varepsilon$ The thesis follows by (A-E).

Case: $l = x{:}l'$ Since l is positive, we have $x > 0$ and l' is positive, hence $\mathsf{allPos}(l') \in \mathcal{S}^{>0}$ and then the thesis follows by rule (A-T).

2.5 Continuity and iteration by rules

At the end of Section 2.2 we have provided an iterative characterisation of least and greatest fixed points of monotone functions. Since we have proved that the inductive and the coinductive interpretations of an inference system coincide with least and greatest fixed points, respectively, of the associated inference operator, which is monotone, such iterative characterisation applies also in this setting. More precisely, by Theorems 2.14, 2.23 and 2.24, we get that, for any inference system $\mathcal{I}$ on a universe $\mathcal{U}$, if the following hold:

- if $F_{\mathcal{I}}$ is upward ω-continuous, then $\mu[\![\mathcal{I}]\!] = \bigcup_{n\in\mathbb{N}} F^n_{\mathcal{I}}(\emptyset)$, and
- if $F_{\mathcal{I}}$ is downward ω-continuous, then $\nu[\![\mathcal{I}]\!] = \bigcap_{n\in\mathbb{N}} F^n_{\mathcal{I}}(\mathcal{U})$.

In this section we will show that there are inference systems for which the inference operator is not (upward or downward) ω-continuous and we will provide sufficient conditions ensuring continuity.

Let us start from upward continuity. First of all, we show an example of inference system whose inference operator is not upward ω-continuous.

EXAMPLE 2.29 : Consider the universe $\mathcal{U} = \mathbb{N} + \{\infty\}$ and the inference system defined by the following rules:

$$\frac{}{0} \qquad \frac{n}{n+1} \qquad \frac{\mathbb{N}}{\infty}$$

where $n \in \mathbb{N}$. For any $k \in \mathbb{N}$, we have $F^k_{\mathcal{I}_1}(\emptyset) = \{0, \ldots, k-1\}$, hence $\bigcup_{k\in\mathbb{N}} F^k_{\mathcal{I}_1}(\emptyset) = \mathbb{N}$. However, due to the last rule, we also have $F_{\mathcal{I}_1}(\mathbb{N}) = \mathbb{N} \cup \{\infty\} \neq \mathbb{N}$, hence $F_{\mathcal{I}_1}$ is not upward ω-continuous. Note also that $\mu[\![\mathcal{I}_1]\!]$ is equal to $\mathbb{N} \cup \{\infty\}$, hence it differs from $\bigcup_{k\in\mathbb{N}} F^k_{\mathcal{I}_1}(\emptyset)$.

Here the problematic rule is the last one, because, having infinitely many premises, it is applicable only after infinitely many iterations, thus breaking the continuity condition. Actually, this fact is not incidental: the absence of rules with infinitely many premises is sufficient to ensure upward ω-continuity of the inference operator. Formally, we have the following results.

DEFINITION 2.30 : An inference syste $\mathcal{I}$ is *finitary* if for all rules $\langle Pr, c\rangle \in \mathcal{I}$, Pr is finite.

THEOREM 2.31 : If $\mathcal{I}$ is finitary, then $F_{\mathcal{I}}$ is upward ω-continuous.

Proof: Consider an increasing ω-chain $(X_n)_{n\in\mathbb{N}}$ of subsets of $\mathcal{U}$ and set $X = \bigcup_{n\in\mathbb{N}} X_n$. Since $F_{\mathcal{I}}$ is monotone, we have $\bigcup_{n\in\mathbb{N}} F_{\mathcal{I}}(X_n) \subseteq F_{\mathcal{I}}(X)$. On the other hand, if $c \in F_{\mathcal{I}}(X)$, then, by definition, there is a rule $\langle Pr, c\rangle \in \mathcal{I}$ such that $Pr \subseteq X$, namely, for all $j \in Pr$, there is $n_j \in \mathbb{N}$ such that $j \in X_{n_j}$. Since Pr is finite by hypothesis, $k = \max\{n_j \mid j \in Pr\}$ is a natural number and, since $(X_n)_{n\in\mathbb{N}}$ is increasing and $n_j \leq k$ for all $j \in Pr$, we get $Pr \subseteq X_k$, hence $c \in F_{\mathcal{I}}(X_k)$. Therefore, we get $F_{\mathcal{I}}(X) \subseteq \bigcup_{n\in\mathbb{N}} F_{\mathcal{I}}(X_n)$ and this implies the thesis. □

COROLLARY 2.32 : If $\mathcal{I}$ is finitary, then

$$\mu[\![\mathcal{I}]\!] = \bigcup_{n\in\mathbb{N}} F^n_{\mathcal{I}}(\emptyset)$$

Proof: Immediate by Theorems 2.14, 2.23 and 2.31 □

This is only a sufficient condition, that is, there are non-finitary inference systems whose inference operator is upward ω-continuous. Furthermore, as we will see in Chapter 8, the inference operator of a finitary inference system enjoys a much stronger continuity property, which will be essential in that case.

We now focus on downward ω-continuity. As above, we first provide an example of an inference system whose inference operator is not downward ω-continuous.

EXAMPLE 2.33 : Consider the universe $\mathcal{U} = \mathbb{N} + \{\infty\}$ and the inference system defined by the following rules:

$$\frac{n}{n+1} \qquad \frac{n}{\infty}$$

where $n \in \mathbb{N}$. For any $k \in \mathbb{N}$, we have $F^k_{\mathcal{I}_2}(\mathcal{U}) = \{k \in \mathbb{N} \mid k \geq n\} \cup \{\infty\}$, hence $\bigcap_{k\in\mathbb{N}} F^k_{\mathcal{I}_2}(\mathcal{U}) = \{\infty\}$. However, we also have $F_{\mathcal{I}_2}(\{\infty\}) = \emptyset \neq \{\infty\}$, hence $F_{\mathcal{I}_2}$ is not downward ω-continuous. Note also that $\nu[\![\mathcal{I}_2]\!]$ is equal to $\emptyset$, hence it differs from $\bigcap_{k\in\mathbb{N}} F^k_{\mathcal{I}_2}(\mathcal{U})$.

Here the problem is due to the fact that the element ∞ is the conclusion of infinitely many rules (one for each natural number), which become all inapplicable only after infinitely many iterations. Actually, this fact is not incidental: the fact that each judgement is the conclusion of only finitely many rules is sufficient to ensure downward ω-continuity of the inference operator. Formally, we have the following results.

DEFINITION 2.34 : An inference system $\mathcal{I}$ on $\mathcal{U}$ is *cofinitary* if, for all $j \in \mathcal{U}$, the set $\{Pr \subseteq \mathcal{U} \mid \langle Pr, j\rangle \in \mathcal{I}\}$ is finite.

THEOREM 2.35 : If $\mathcal{I}$ is cofinitary, then $F_{\mathcal{I}}$ is downward ω-continuous.

Proof: Consider a decreasing ω-chain $(X_n)_{n\in\mathbb{N}}$ and set $X = \bigcap_{n\in\mathbb{N}} X_n$. Since $F_{\mathcal{I}}$ is monotone, we have $F_{\mathcal{I}}(X) \subseteq \bigcap_{n\in\mathbb{N}} F_{\mathcal{I}}(X_n)$. On the other hand, suppose $c \in F_{\mathcal{I}}(X_n)$ for all $n \in \mathbb{N}$, and set $A_c = \{\Pi \subseteq \mathcal{U} \mid \langle \Pi, c\rangle \in \mathcal{I}\}$, $n_\Pi = \sup\{n \in \mathbb{N} \mid \Pi \subseteq X_n\}$, for each $\Pi \in A_c$, and $n_c = \sup\{n_\Pi \mid \Pi \in A_c\}$. If n_c were a natural number, say m, then we would have $n_\Pi \leq m$, for all $\Pi \in A_c$, and so $\Pi \nsubseteq X_{m+1}$, for each $\Pi \in A_c$, because, if $\Pi \subseteq X_{m+1}$, then $n_\pi \geq m + 1 > m$, which is not possible. Hence, n_c is not a natural number, that is, $n_c = \infty$. Then, since A_c is finite by hypothesis and n_c is infinite, there exists $\Pi \in A_c$ such that n_Π is infinite, hence, for all $n \in \mathbb{N}$, there is $k_n \geq n$ such that $\Pi \subseteq X_{k_n}$. Because $(X_n)_{n\in\mathbb{N}}$ is decreasing, for all $n \in \mathbb{N}$, as $k_n \geq n$, we have $\Pi \subseteq X_{k_n} \subseteq X_n$. Therefore, $\Pi \subseteq X$ and so $c \in F_{\mathcal{I}}(X)$, as needed. □

COROLLARY 2.36 : If $\mathcal{I}$ is cofinitary, then

$$\nu[\![\mathcal{I}]\!] = \bigcap_{n\in\mathbb{N}} F^n_{\mathcal{I}}(\mathcal{U})$$

Proof: Immediate by Theorems 2.14, 2.24 and 2.35. □

Also in this case, this is only a sufficient condition, that is, there are non-cofinitary inference systems whose inference operator is downward ω-continuous.

3

Inference systems with corules

Inference systems are a widespread and versatile framework to define possibly recursive judgements: they provide solid and fairly simple foundations for both inductive and coinductive reasoning both in proof-theoretic and in model-theoretic terms. More precisely, as we have seen, given a set of rules, one has these two possibilities: either taking the smallest possible interpretation, that is, the inductive one, or taking the largest possible interpretation, that is, the coinductive one.

This strong dichotomy between inductive and coinductive interpretation makes the framework a bit rigid, because it allows us to choose only between two possibilities, while there is a variety of intermediate interpretations, lying between the smallest and the largest ones, which cannot be selected. Furthermore, there are cases where neither the inductive nor the coinductive one are able to provide the expected meaning of an inference system, because it is indeed one of such intermediate interpretations.

Let us illustrate this issue on some simple examples dealing with judgements on lists of integers. We start by considering simple variations of judgements defined in Section 2.1. Let $\mathbb{B} = \{\mathrm{T}, \mathrm{F}\}$ be the set of truth values, consider the judgements $\mathsf{member}_{\mathbb{B}}(x, l, b)$ and $\mathsf{allPos}_{\mathbb{B}}(l, b)$ with $b \in \mathbb{B}$ such that

- $\mathsf{member}_{\mathbb{B}}(x, l, \mathrm{T})$ holds iff $\mathsf{member}(x, l)$ holds, and otherwise $\mathsf{member}_{\mathbb{B}}(x, l, \mathrm{F})$ holds
- $\mathsf{allPos}_{\mathbb{B}}(l, \mathrm{T})$ holds iff $\mathsf{allPos}(l)$ holds, and otherwise $\mathsf{allPos}_{\mathbb{B}}(l, \mathrm{F})$ holds.

We can define these judgements by means of the following inference systems:

$$\frac{}{\mathsf{member}_{\mathbb{B}}(x, \varepsilon, \mathrm{F})} \qquad \frac{}{\mathsf{member}_{\mathbb{B}}(x, x{:}l, \mathrm{T})} \qquad \frac{\mathsf{member}_{\mathbb{B}}(x, l, b)}{\mathsf{member}_{\mathbb{B}}(x, y{:}l, b)}\ x \neq y$$

$$\frac{}{\mathsf{allPos}_{\mathbb{B}}(\varepsilon, \mathrm{T})} \qquad \frac{}{\mathsf{allPos}_{\mathbb{B}}(x{:}l, \mathrm{F})}\ x \leq 0 \qquad \frac{\mathsf{allPos}_{\mathbb{B}}(l, b)}{\mathsf{allPos}_{\mathbb{B}}(x{:}l, b)}\ x > 0$$

For both definitions, neither the inductive interpretation, nor the coinductive one works well on infinite lists. For the judgement $\mathsf{member}_{\mathbb{B}}(x, l, b)$, with the inductive interpretation we cannot derive any judgement of shape $\mathsf{member}_{\mathbb{B}}(x, l, \mathrm{F})$ where l is an infinite list and x does not belong to l, while with the coinductive interpretation we get both $\mathsf{member}_{\mathbb{B}}(x, l, \mathrm{F})$ and $\mathsf{member}_{\mathbb{B}}(x, l, \mathrm{T})$. For the judgement $\mathsf{allPos}_{\mathbb{B}}(l, b)$, with the inductive interpretation we cannot derive any judgement of shape $\mathsf{allPos}_{\mathbb{B}}(l, \mathrm{T})$ where l is an infinite list containing only positive elements, while with the coinductive interpretation we get both

$$\dfrac{\dfrac{\vdots}{\mathsf{member}_{\mathbb{B}}(2, 1{:}1{:}1{:}\ldots, \textsc{f})}}{\mathsf{member}_{\mathbb{B}}(2, 1{:}1{:}1{:}\ldots, \textsc{f})} \qquad \dfrac{\dfrac{\vdots}{\mathsf{allPos}_{\mathbb{B}}(1{:}1{:}1{:}\ldots, \textsc{t})}}{\mathsf{allPos}_{\mathbb{B}}(1{:}1{:}1{:}\ldots, \textsc{t})} \qquad \dfrac{\dfrac{\vdots}{\mathsf{maxElem}(1{:}1{:}1{:}\ldots, 1)}}{\mathsf{maxElem}(1{:}1{:}1{:}\ldots, 1)}$$

$$\dfrac{\dfrac{\vdots}{\mathsf{member}_{\mathbb{B}}(2, 1{:}1{:}1{:}\ldots, \textsc{t})}}{\mathsf{member}_{\mathbb{B}}(2, 1{:}1{:}1{:}\ldots, \textsc{t})} \qquad \dfrac{\dfrac{\vdots}{\mathsf{allPos}_{\mathbb{B}}(1{:}1{:}1{:}\ldots, \textsc{f})}}{\mathsf{allPos}_{\mathbb{B}}(1{:}1{:}1{:}\ldots, \textsc{f})} \qquad \dfrac{\dfrac{\vdots}{\mathsf{maxElem}(1{:}1{:}1{:}\ldots, 2)}}{\mathsf{maxElem}(1{:}1{:}1{:}\ldots, 2)}$$

FIGURE 3.1 Some infinite derivations for judgements $\mathsf{member}_{\mathbb{B}}(x, l, b)$, $\mathsf{allPos}_{\mathbb{B}}(l,)$ and $\mathsf{maxElem}(l, x)$.

$\mathsf{allPos}_{\mathbb{B}}(l, \textsc{t})$ and $\mathsf{allPos}_{\mathbb{B}}(l, \textsc{f})$. Some examples of derivations of unexpected judgements can be found in the bottom section of Figure 3.1.

We consider now another example, defining the predicate $\mathsf{maxElem}(l, x)$ stating that x is the maximum of the list l. The definition is given by the following inference system

$$\frac{}{\mathsf{maxElem}(x{:}\varepsilon, x)} \qquad \frac{\mathsf{maxElem}(l, y)}{\mathsf{maxElem}(x{:}l, z)}\ z = \max\{x, y\}$$

The inductive interpretation works well on finite lists, but does not allow to derive any judgement on infinite lists, again, because, to compute a maximum, we need to inspect the whole list. The coinductive interpretation still works well on finite lists, but, again, we can derive too many judgements regarding infinite lists: for instance, if $l = 1{:}1{:}1{:}\ldots$ is the infinite list of 1s, as depicted in Figure 3.1, we can derive both $\mathsf{maxElem}(l, 1)$, which is correct, and $\mathsf{maxElem}(l, 2)$, that is clearly wrong, since 2 does not belong to l.

In all these examples, the inductive interpretation is too restrictive, as expected, while, more surprisingly, the coinductive one allows the derivation of too many judgements. Hence, to get the intended semantics, we need a way to select a middleway interpretation. In this chapter, we present a generalisation of inference systems which makes this possible, relying on *corules*.

Corules are special rules that need to be provided together with standard rules in order to tune their semantics. More precisely, they allow us to refine the coinductive interpretation of standard rules, removing some undesired judgements, thus obtaining an interpretation which is neither the smallest nor the largest one. For instance, in all the above three examples, the coinductive interpretation is undetermined on infinite lists (we can derive both correct and incorrect judgements); but, as we will see, adding suitable corules we can remove all incorrect judgements, thus obtaining the correct interpretation. An important property is that standard inductive and coinductive interpretations are particular cases, that is, they can be recovered by specific choices of corules, thus justifying why this framework is said to be a generalisation of standard inference systems.

The notion of corules, originally introduced by Ancona, Dagnino, and Zucca (2017b) and Dagnino (2019), has been inspired by some operational models for programming languages supporting corecursion, *e.g.*, (Ancona and Zucca, 2012, 2013; Ancona, 2013) and, in our intention, this generalisation of inference systems will serve as an abstract framework for a better understanding and formalisation of such operational models, as we have done in (Ancona et al., 2020b; Dagnino, Ancona, and Zucca, 2020) and in the last chapter of this thesis.

The rest of the chapter is organized as follows. Section 3.1 introduces corules by several examples and define the interpretation generated by corules. Section 3.2 defines the bounded fixed point in a lattice-theoretic setting and proves that the interpretation generated by corules coincides with such fixed point of the inference operator. Section 3.3 presents several equivalent proof-theoretic characterisations of the interpretation generated by corules. Finally, Section 3.4 discuss proof techniques for corules and Section 3.5 show how to use such techniques by some more involved examples.

3.1 A gentle introduction: definitions and examples

In this section we introduce our generalization of inference systems, discussing it on some examples. Let us start from the fundamental definition:

DEFINITION 3.1 : An *inference system with corules*, or *generalised inference system*, is a pair of inference systems $\langle \mathcal{I}, \mathcal{I}_{co} \rangle$, where elements of $\mathcal{I}$ are called *rules*, while elements of $\mathcal{I}_{co}$ are called *corules*.

A corule $\langle Pr, c \rangle \in \mathcal{I}_{co}$ will be also written as $\dfrac{Pr}{c}$, like standard rules, but with a thicker line. Corules are very much like standard rules, but will be used in a special way, to refine the coinductive interpretation of $\mathcal{I}$. Basically, corules impose additional conditions that infinite derivations has to satisfy, thus providing a finer control on them.

Let us start with an introductory example concerning graphs, that are a widely used non-well-founded data type. Recall that (cf. p.18) a graph G is modelled by the adjacency function $G : V \to \wp(V)$, where V is the set of nodes, that is, for each node $v \in V$, $G(v)$ is the set of nodes adjacent to v. We define the judgement $v \to^\star N$ stating that N is the set of nodes reachable from v, by the following rules and corules:

$$\frac{v_1 \to^\star N_1 \quad \ldots \quad v_k \to^\star N_k}{v \to^\star \{v\} \cup N_1 \cup \ldots \cup N_k}\ G(v) = \{v_1, \ldots, v_k\} \qquad \frac{}{v \to^\star \emptyset}$$

To be more concrete, we consider the graph drawn in Figure 3.2, whose corresponding rules are reported in the same figure.

Let us ignore for a moment corules and reason about the standard interpretations. It is clear that, if we interpret the system inductively, we will only prove the judgement $c \to^\star \{c\}$, because it is the only axiom and other rules do not

$$\frac{b \to^\star N}{a \to^\star \{a\} \cup N} \qquad \frac{a \to^\star N}{b \to^\star \{b\} \cup N} \qquad \frac{}{c \to^\star \{c\}}$$

$$\frac{}{a \to^\star \emptyset} \qquad \frac{}{b \to^\star \emptyset} \qquad \frac{}{c \to^\star \emptyset}$$

FIGURE 3.2 Rules defining $v \to^\star N$ on a concrete graph

$$\dfrac{\dfrac{\dfrac{\vdots}{a \to^\star \{a,b\}}}{b \to^\star \{a,b\}}}{a \to^\star \{a,b\}} \qquad \dfrac{\dfrac{\dfrac{\vdots}{b \to^\star \{a,b\}}}{a \to^\star \{a,b\}}}{b \to^\star \{a,b\}} \qquad \dfrac{\dfrac{\dfrac{\vdots}{a \to^\star \{a,b,c\}}}{b \to^\star \{a,b,c\}}}{a \to^\star \{a,b,c\}} \qquad \dfrac{\dfrac{\dfrac{\vdots}{b \to^\star \{a,b,c\}}}{a \to^\star \{a,b,c\}}}{b \to^\star \{a,b,c\}}$$

FIGURE 3.3 Some infinite derivations of $v \to^\star N$ for the graph in Figure 3.2.

depend on it. In other words, the judgement $v \to^\star N$, like other judgements on graphs, cannot be defined inductively by structural recursion, since the structure is not well-founded. In particular, the problem are cycles, where the proof may be "trapped", continuously unfolding the structure of the graph without ever reaching a base case. Usual inductive approaches to visits on graphs rely on some auxiliary structure, used to mark already visited nodes. In this way, we avoid visiting twice the same node, thus breaking cycles and solving this issue.

On the other hand, if we interpret the meta-rules coinductively (excluding again the corules), then we get the correct judgements $a \to^\star \{a,b\}$ and $b \to^\star \{a,b\}$, but we also get the wrong judgements $a \to^\star \{a,b,c\}$ and $b \to^\star \{a,b,c\}$, as shown by derivations in Figure 3.3.

As already said, corules allow us to impose additional conditions on derivations to be considered correct, thus providing us with a tool for disregarding undesired derivations. More precisely, cf. Definition 3.2, we are allowed to build arbitrary (well-founded or not) derivations, but using only judgements that can be derived by a finite derivation using also corules. Hence, we can use coinduction, namely, non-well-founded derivations, but we restrict it by using corules.

For instance, considering derivations in Figure 3.3, the first two derivations, proving the judgements $a \to^\star \{a,b\}$ and $b \to^\star \{a,b\}$, are correct, because they use only judgements having a finite derivation using also corules, as

shown below.

$$\dfrac{\dfrac{\dfrac{}{a \to^\star \emptyset}}{b \to^\star \{b\}}}{a \to^\star \{a, b\}} \qquad \dfrac{\dfrac{\dfrac{}{b \to^\star \emptyset}}{a \to^\star \{a\}}}{b \to^\star \{a, b\}}$$

On the other hand, the last two derivations in Figure 3.3, proving judgements $a \to^\star \{a, b, c\}$ and $b \to^\star \{a, b, c\}$, have no finite proof tree using also corules, because c is not reachable from a, hence, they are not correct, as expected.

More generally, given an inference system with corules $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$, we can construct its semantics by the following two steps:

- first we take the inductive interpretation of $\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}$, $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$, which is the set of judgements having a finite proof tree in $\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}$,
- then, we take the coinductive interpretation of $\mathcal{I}$ restricted to rules with conclusion in $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$.

In other words, we are allowed to use arbitrary (well-founded or not) derivations in $\mathcal{I}$, but built using only judgements in $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$. Note that, since well-founded proof trees in $\mathcal{I}$ are also well-founded proof trees in $\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}$, this additional condition is non-trivial only for non-well-founded proof trees in $\mathcal{I}$, because judgements occurring in a well-founded proof tree in $\mathcal{I}$ automatically belong to $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$.

Formally, we have the following definition, where $\mathcal{I}_{|X}$, for $X \subseteq \mathcal{U}$, denotes the inference system $\{\langle Pr, c\rangle \in \mathcal{I} \mid c \in X\}$:

DEFINITION 3.2 : The interpretation of an inference system with corules $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$ is the set $\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] = \nu[\![\mathcal{I}_{|\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]}]\!]$.

In the following we will write $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle \vdash_\nu j$ for $j \in \nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$.

If we consider again the example of the graph in Figure 3.2, the semantics is constructed as follows. In the first step, we obtain the following set of judgements:

$$\begin{aligned} A = \{&a \to^\star \emptyset, b \to^\star \emptyset, c \to^\star \emptyset, c \to^\star \{c\}, a \to^\star \{a\}, b \to^\star \{b\}, \\ &a \to^\star \{a, b\}, b \to^\star \{a, b\}\} \end{aligned}$$

which contains all the judgements having a finite proof tree using also corules.

For the second step, first of all we have to construct the inference system $\mathcal{I}_{|\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]}$, whose rules are those of $\mathcal{I}$ (in Figure 3.2) with conclusion in A, described above. Hence, they are the following:

$$\dfrac{b \to^\star N}{a \to^\star \{a\} \cup N}\ c \notin N \qquad \dfrac{a \to^\star N}{b \to^\star \{b\} \cup N}\ c \notin N \qquad \dfrac{}{c \to^\star \{c\}}$$

These rules have to be interpreted coinductively, hence we get the following set

$$B = \{c \to^\star \{c\}, a \to^\star \{a, b\}, b \to^\star \{a, b\}\}$$

which is the expected semantics.

As another example, we consider the definition of the first-sets in a context-free grammar. Let us represent a context-free grammar by its set of terminals

T, its set of non-terminals N, and all the productions $A ::= \beta_1 \mid \ldots \mid \beta_n$ for each non-terminal $A \in N$. Recall that, for each $\alpha \in (T \cup N)^+$, we can define the set $\mathit{first}(\alpha) = \{\sigma \in T \mid \alpha \rightarrow^\star \sigma\beta \text{ for some } \beta \in (T \cup N)^+\}$. Informally, $\mathit{first}(\alpha)$ is the set of the initial terminal symbols of the strings which can be derived from a string α in 0 or more steps.

We define the judgement $\mathsf{first}(\alpha, F)$ by the following inference system with corules, where $F \subseteq T$.

$$\frac{}{\mathsf{first}(\sigma\alpha, \{\sigma\})}\ \sigma \in T \qquad \frac{}{\mathsf{first}(\epsilon, \emptyset)} \qquad \frac{}{\mathsf{first}(A, \emptyset)}\ A \in N$$

$$\frac{\mathsf{first}(A, F)}{\mathsf{first}(A\alpha, F)}\ \begin{array}{l} A \in N \\ A \twoheadrightarrow^\star \epsilon \end{array} \qquad \frac{\mathsf{first}(A, F) \qquad \mathsf{first}(\alpha, F')}{\mathsf{first}(A\alpha, F \cup F')}\ \begin{array}{l} A \in N \\ A \rightarrow^\star \epsilon \end{array}$$

$$\frac{\mathsf{first}(\beta_1, F_1) \qquad \cdots \qquad \mathsf{first}(\beta_n, F_n)}{\mathsf{first}(A, F_1 \cup \cdots \cup F_n)}\ A ::= \beta_1 \mid \ldots \mid \beta_n$$

The rules of the inference system correspond to the natural recursive definition of first-sets. Note, in particular, that in a string of shape $A\alpha$, if the non-terminal A is *nullable*, that is, we can derive from it the empty string, then the first-set of $A\alpha$ should also include the first-set for α.

As in the previous example on graphs, the problem with this recursive definition is that, since the non-terminals in a grammar can mutually refer to each other, the predicate defined by the inductive interpretation can be undefined, since it may never reach a base case. From another perspective, this means that a naive recursive top-down implementation might not terminate. For this reason, first-sets are typically computed by an imperative bottom-up algorithm, or the top-down implementation is corrected by marking already encountered non-terminals, analogously to what is done for visiting graphs. Again as in the previous example, the coinductive interpretation may be undetermined, allowing the derivation also of wrong judgements, whereas, with the corules, we get the expected result.

Let us now consider some examples of judgements on arbitrary (finite or infinite) lists of integers. Using corules, we can get the right semantics for the examples discussed in the introduction of this chapter, namely, definitions for judgements $\mathsf{member}_{\mathbb{B}}(x, l, b)$, $\mathsf{allPos}_{\mathbb{B}}(l, b)$ and $\mathsf{maxElem}(l, x)$. We report below the inference systems with corules defining these judgements.

$$\frac{}{\mathsf{member}_{\mathbb{B}}(x, \varepsilon, \mathrm{F})} \qquad \frac{}{\mathsf{member}_{\mathbb{B}}(x, x{:}l, \mathrm{T})}$$

$$\frac{\mathsf{member}_{\mathbb{B}}(x, l, b)}{\mathsf{member}_{\mathbb{B}}(x, y{:}l, b)}\ x \neq y \qquad \frac{}{\mathsf{member}_{\mathbb{B}}(x, l, \mathrm{F})}$$

$$\frac{}{\mathsf{allPos}_{\mathbb{B}}(\varepsilon, \mathrm{T})} \qquad \frac{}{\mathsf{allPos}_{\mathbb{B}}(x{:}l, \mathrm{F})}\ x \leq 0$$

$$\frac{\mathsf{allPos}_{\mathbb{B}}(l, b)}{\mathsf{allPos}_{\mathbb{B}}(x{:}l, b)}\ x > 0 \qquad \frac{}{\mathsf{allPos}_{\mathbb{B}}(l, \mathrm{T})}$$

$$\frac{}{\mathsf{maxElem}(x{:}\varepsilon, x)} \qquad \frac{\mathsf{maxElem}(l, y)}{\mathsf{maxElem}(x{:}l, z)}\, z = \max\{x, y\} \qquad \frac{}{\mathsf{maxElem}(x{:}l, x)}$$

As shown in Figure 3.1, the standard coinductive interpretation of rules allows the derivation of too many judgements. Corules impose additional conditions on infinite derivations to be considered valid: they can only use judgements having a finite proof tree built using also corules. We can spell out these additional conditions for these judgements as follows:

- $\mathsf{member}_{\mathbb{B}}(x, l, b)$ can occur in an infinite derivation iff $b = \mathrm{F}$,
- $\mathsf{allPos}_{\mathbb{B}}(l, b)$ can occur in an infinite derivation iff $b = \mathrm{T}$, and
- $\mathsf{maxElem}(l, x)$ can occur in an infinite derivation iff x belongs to l.

Therefore, introducing corules, all infinite derivations in the bottom section of Figure 3.1 are incorrect, because they use judgements that do not satisfy these conditions. More generally, it can be proved that, through corules, we get the expected semantics.

A similar example is given by the judgement $\mathsf{elems}(l, X)$, stating that X is the carrier of the list l, that is, the set of all elements appearing in l. This judgement can be defined using corules as follows:

$$\frac{}{\mathsf{elems}(\varepsilon, \emptyset)} \qquad \frac{\mathsf{elems}(l, X)}{\mathsf{elems}(x{:}l, \{x\} \cup X)} \qquad \frac{}{\mathsf{elems}(l, \emptyset)}$$

If we ignore the corule and interpret the system coinductively, then we can prove $\mathsf{elems}(l, X)$ for any superset X of the carrier of l, if l is infinite. Corules again allow us to filter out undesired derivations. For instance, for $l = 1{:}1{:}1{:}\ldots$ the infinite list of 1s, any judgement $\mathsf{elems}(l, X)$ with $1 \in X$ can be derived. Indeed, for any such judgement, we can construct an infinite proof tree by applying infinitely many times the last rule. With corules, instead, we only consider the infinite trees built by judgements having a finite proof tree using also corules. This is only true for $X = \{1\}$.

In next sections, we will study properties of $\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$, notably, we will show that it is indeed a fixed point of the inference operator $F_{\mathcal{I}}$ as expected (cf. Section 3.2), hence an interpretation of $\mathcal{I}$ according to Definition 2.2. Such a fixed point will be constructed by taking the greatest consistent subset of the inductive interpretation of $\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}$. Then, we will also provide several proof-theoretic characterisations (cf. Section 3.3), making formal ideas described in this section.

3.2 Fixed point semantics for corules

According to Definition 2.2, an interpretation of an inference system $\mathcal{I}$ on a universe $\mathcal{U}$ is a closed and consistent subset of $\mathcal{U}$, or equivalently, as described in Section 2.3, a fixed point of the inference operator $F_{\mathcal{I}}$ associated with $\mathcal{I}$. We aim at showing that this property holds for $\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$, hence it is indeed an interpretation of $\mathcal{I}$.

In this section, we will develop the theory needed for this result and some important consequences. In order to construct the fixed point we need, as done for the least and greatest fixed points in Chapter 2, we work in the general framework of *lattice theory* (cf. Davey and Priestley, 2002), so that we can highlight only the essential structure. More precisely, in Section 3.2.1 we define the *bounded fixed point*, showing in Section 3.2.2 it corresponds to the interpretation of an inference system with corules as defined in Definition 3.2, and it subsumes both inductive and coinductive interpretations.

3.2.1 The bounded fixed point

Let us assume a complete lattice $\langle L, \sqsubseteq \rangle$ and monotone functions $F : L \to L$ and $G : L \to L$. We refer to Section 2.2 for basic notions of lattice theory.

Let us introduce some notations: $F \sqcup G : L \to L$ is the function defined by $x \mapsto F(x) \sqcup G(x)$, and, for all $z \in L$, $F_{\sqcap z} : L \to L$ is the function defined by $x \mapsto F(x) \sqcap z$. Further, we define the *interior* of F as the function $\mathbf{int}_F : L \to L$ given by

$$\mathbf{int}_F(x) = \bigsqcup \{z \in \mathsf{post}(F) \mid z \sqsubseteq x\}$$

In other words, $\mathbf{int}_F(x)$ is the greatest post-fixed point of F below x. Functions $F \sqcup G$, $F_{\sqcap z}$, for all $z \in L$, and $\mathbf{int}_F$ are all trivially monotone. Then we can define the *bounded fixed point* of F generated by G as follows:

DEFINITION 3.3 (Bounded fixed point): The *bounded fixed point of F generated by G*, denoted by $\nu[F, G]$, is defined by

$$\nu[F, G] = \nu F_{\sqcap \mu(F \sqcup G)}$$

Expanding the above definition, using the Knaster-Tarski theorem (cf. Theorem 2.10), we get the following corollary:

COROLLARY 3.4 : $\nu[F, G] = \mathbf{int}_F(\mu(F \sqcup G))$.

Therefore, the bounded fixed point is constructed in two steps:

- first, we take the least fixed point of $F \sqcup G$, and
- then, we take the greatest post-fixed point of F below it.

As for standard least and greatest fixed points, from this observation we immediately get the following property of the bounded fixed point: let $x \in L$, then

$(\nu_{\mathrm{B}}\mathrm{P})$ if $x \sqsubseteq F(x)$ and $x \sqsubseteq \mu(F \sqcup G)$, then $x \sqsubseteq \nu[F, G]$.

which is basically the same as $(\nu\mathrm{P})$ (cf. p.21), but with the additional constraint, named *boundedness* condition, requiring x to be below $\mu(F \sqcup G)$.

We now have to check that the above definition is a good definition, that is, we need to show that $\nu[F, G]$ is indeed a fixed point of F. To this end, we rely on the following fact:

PROPOSITION 3.5 : Let $z \in L$. If z is a pre-fixed point of F, then $\nu F_{\sqcap z}$ is a fixed point of F.

Proof: Let $x = \nu F_{\sqcap z}$, hence, by definition, we have $x = F(x) \sqcap z$ and so $x \sqsubseteq z$. By Corollary 3.4 and Proposition 2.9 (2), we know that x is a post-fixed point of F. To check the other inequality, just note the following:

$$\begin{aligned} F(x) &= F(x \sqcap z) && x \sqsubseteq z \\ &\sqsubseteq F(x) \sqcap F(z) && F \text{ is monotone} \\ &\sqsubseteq F(x) \sqcap z && F(z) \sqsubseteq z \\ &= x && x \text{ fixed point of } F_{\sqcap z} \end{aligned}$$

This result can also be obtained by applying the Knaster-Tarski theorem (cf. Theorem 2.10) to the function F_{L_z} obtained by restricting F to the sublattice $L_z = \{y \in L \mid y \sqsubseteq z\}$. F_{L_z} is well-defined as F is monotone and z is pre-fixed. □

PROPOSITION 3.6 : $\nu[F, G]$ is a fixed point of F.

Proof: Let $z = \mu(F \sqcup G)$, hence we have $F(z) \sqsubseteq F(z) \sqcup G(z) = z$. Hence, the thesis follows from Proposition 3.5. □

Note that, to get that the bounded fixed point is well-defined, it is essential that the bound $\mu(F \sqcup G)$ is a pre-fixed point of F, otherwise we are not guaranteed to obtain a fixed point of F. This is the reason why, in the first step of this construction, we *cannot* just take μG as bound, because, in general, it is not a pre-fixed point of F; for instance, if G maps any element to $\bot$ (the bottom element of L) and F maps any element to $x \neq \bot$, clearly $\mu G = \bot \neq x = F(\mu G)$. However, the first step is not enough, because, in general, $\mu(F \sqcup G)$ is *not* a fixed point of F: we need the two steps to obtain the expected result.

Note also that the definition of bounded fixed point is asymmetric, that is, we take the greatest fixed point bounded from above by a least fixed point, rather than the other way round. This is motivated by the fact that, as explained in Section 3.1, we essentially need a greatest fixed point, but we want to "constrain" it in some way . Investigating the dual construction is a matter of further work.

We now discuss some properties of the bounded fixed point. In the following, for all $z \in L$, we denote by $K_z : L \to L$ the constant function mapping any $x \in L$ to z. This function is obviously monotone. We also write $F_1 \sqsubseteq F_2$, for F_1 and F_2 function on L, when, for all $x \in L$, $F_1(x) \sqsubseteq F_2(x)$.

PROPOSITION 3.7 : The following hold:

1. If $z \in L$ is a fixed point of F, then $\nu[F, K_z] = z$.
2. For all $G_1, G_2 : L \to L$, if $G_1 \sqsubseteq G_2$, then $\nu[F, G_1] \sqsubseteq \nu[F, G_2]$.

Proof:

1. We have, by hypothesis, $F(z) \sqcup K_z(z) = z \sqcup z = z$ and $(F \sqcup K_z)(x) \sqsubseteq x$ implies $z \sqsubseteq F(x) \sqcup z = (F \sqcup K_z)(x) \sqsubseteq x$, hence $\mu(F \sqcup K_z) = z$. Then, by Definition 3.3, we know that $\nu[F, K_z] \sqsubseteq z$ and, as $z \sqsubseteq F(z)$ by hypothesis, we get $z \sqsubseteq \nu[F, K_z]$ from Corollary 3.4, which proves the thesis.
2. It is easy to check by (μP) (cf. p21) that $\mu(F \sqcup G_1) \sqsubseteq \mu(F \sqcup G_2)$, hence we get the thesis by Corollary 3.4 and monotonicity of $\mathbf{int}_F$. □

Therefore, by Proposition 3.6 we already know that $\nu[F, G]$ is a fixed point for any $G : L \to L$ and, on the other hand, Proposition 3.7 (1) says that all fixed points of F can be generated as bounded fixed points. In other words, considering $\nu[F, -]$ as a function from the poset of monotone endofunctions on L to L itself, Proposition 3.7 (1) implies that the range of this function is exactly $\mathsf{fix}(F)$. Moreover, Proposition 3.7 (2) states that $\nu[\![F, -]\!]$ is a monotone function.

An important fact is that least and greatest fixed points can be retrieved as bounded fixed points for particular choices of G, as stated in the following proposition.

PROPOSITION 3.8 : The following hold:

1. $\nu[F, K_\top] = \nu F$.
2. $\nu[F, K_\bot] = \mu F$.

Proof: Since $\nu[F, K_\bot]$ and $\nu[F, K_\top]$ are fixed points by Proposition 3.6, we have $\mu F \sqsubseteq \nu[F, K_\bot]$ and $\nu[F, K_\top] \sqsubseteq \nu F$. Let $z \in L$ be a fixed point of F, then by Proposition 3.7 (1) we have $z = \nu[F, K_z]$. Since $\bot \sqsubseteq z \sqsubseteq \top$, we have $K_\bot \sqsubseteq K_z \sqsubseteq K_\top$, hence, by Proposition 3.7 (2), we get $\nu[F, K_\bot] \sqsubseteq \nu[F, K_z] \sqsubseteq \nu[F, K_\top]$. This implies $\nu[F, K_\bot] \sqsubseteq \mu F$ (when $z = \mu F$) and $\nu F \sqsubseteq \nu[F, K_\top]$ (when $z = \nu F$), hence we get the thesis. □

We now provide an iterative characterisation of the bounded fixed point, following what described at the end of Section 2.2 for standard least and greatest fixed points. We refer to that section for basic definitions (ω-chains and continuous functions).

PROPOSITION 3.9 : Let $z \in L$ be a pre-fixed point of F. Then

1. for all $n \in \mathbb{N}$, $\mathbf{int}_F(z) = \mathbf{int}_F(F^n(z))$ and,
2. $\mathbf{int}_F(z) = \mathbf{int}_F(\bigsqcap_{n \in \mathbb{N}} F^n(z))$

Proof: Note that, since z is pre-fixed, the sequence $(F^n(z))_{n \in \mathbb{N}}$ is a descending ω-chain, that is, for all $n \in \mathbb{N}$, we have $F^{n+1}(z) \sqsubseteq F^n(z)$, which implies that $F^n(z)$ is a pre-fixed point of F, for all $n \in \mathbb{N}$.

1. We prove the statement by induction on n. The base case ($n = 0$) is trivial. For the induction step, first note that, by definition of $\mathbf{int}_F$, $\mathbf{int}_F(F^n(z))$ is a post-fixed point, that is, $\mathbf{int}_F(F^n(z)) \sqsubseteq F(\mathbf{int}_F(F^n(z)))$, and $\mathbf{int}_F(F^n(z)) \sqsubseteq F^n(z)$, hence, by monotonicity of F, we get $F(\mathbf{int}_F(F^n(z))) \sqsubseteq F^{n+1}(z)$. Now, by transitivity of $\sqsubseteq$, we get $\mathbf{int}_F(F^n(z)) \sqsubseteq F^{n+1}(z)$. Therefore, again by definition of $\mathbf{int}_F$, we conclude $\mathbf{int}_F(F^n(z)) \sqsubseteq \mathbf{int}_F(F^{n+1}(z))$. On the other hand, we have $F^{n+1}(z) \sqsubseteq F^n(z)$ and, by monotonicity of $\mathbf{int}_F$, we get the other inequality, and this implies $\mathbf{int}_F(F^n(z)) = \mathbf{int}_F(F^{n+1}(z))$. Finally, thanks to the induction hypothesis, we get the thesis.

2. By Item 1, we have $\mathbf{int}_F(z) \sqsubseteq F^n(z)$ for all $n \in \mathbb{N}$, hence $\mathbf{int}_F(z) \sqsubseteq \bigsqcap_{n\in\mathbb{N}} F^n(z)$. Therefore, by definition of $\mathbf{int}_F$, we get $\mathbf{int}_F(z) \sqsubseteq \mathbf{int}_F(\bigsqcap_{n\in\mathbb{N}} F^n(z))$. On the other hand, we have $\bigsqcap_{n\in\mathbb{N}} F^n(z) \sqsubseteq z$, hence, by monotonicity of $\mathbf{int}_F$, we get the other inequality, and this implies the thesis. □

Another way to read the above proposition is that, given a pre-fixed point $z \in L$, we obtain the same greatest (post-)fixed point below z if we take as "bound" any element $F^n(z)$ of the descending chain of n-iterations of F. Moreover, Proposition 3.9 (2) says also that we obtain the same greatest (post-)fixed point induced by z if we take as bound the limit (greatest lower bound) of that chain.

We conclude this section extending Theorem 2.14 to the bounded fixed point.

PROPOSITION 3.10 : Let $z \in L$ be a pre-fixed point of F, then, if F is downward ω-continuous, then $\mathbf{int}_F(z) = \bigsqcap_{n\in\mathbb{N}} F^n(z)$.

Proof: The set $L_z = \{x \in L \mid x \sqsubseteq z\}$ is a complete lattice with top element z and the restriction of F on L_z is well-defined, since z is a pre-fixed point, and downward ω-continuous. Therefore, by Proposition 3.5, $\mathbf{int}_F(z)$ is the greatest fixed point of F in L_z, hence, since F is downward ω-continuous, we get the thesis by Theorem 2.14. □

Note that the above proposition, like Theorem 2.14, requires an additional condition of F, that is, it has to be continuous. Under this condition the above result immediately applies to the bounded fixed point, providing us with an iterative characterization of it.

COROLLARY 3.11 : If F is downward ω-continuous, then $\nu[F, G] = \bigsqcap_{n\in\mathbb{N}} F^n(\mu(F \sqcup G))$.

Proof: Since $\mu(F \sqcup G)$ is a pre-fixed point of F and F is downward ω-continuous, the thesis immediately follows from Proposition 3.10. □

3.2.2 Corules as generator

In this part of the section we come back to inference systems and we show that the interpretation of $\langle \mathcal{I}, \mathcal{I}_{co} \rangle$, as in Definition 3.2, is indeed an interpretation

of $\mathcal{I}$, that is, a fixed point of the inference operator associated with $\mathcal{I}$. In Section 3.1 we have described two steps to construct $\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$:

- First, we consider the inference system $\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}$ obtained putting together rules and corules, and we take its inductive interpretation $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$.
- Then, we take the coinductive interpretation of the inference system obtained from $\mathcal{I}$ by keeping only rules with conclusion in $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$, that is, we have

$$\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] = \nu[\![\mathcal{I}_{|\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]}]\!]$$

The definition of the bounded fixed point is the formulation of these two steps in the general setting of complete lattices. Indeed, the inference operators $F_{\mathcal{I}}$ and $F_{\mathcal{I}_{\mathsf{co}}}$, associated with rules and corules, respectively, are monotone functions on the complete lattice $\langle \wp(\mathcal{U}), \subseteq \rangle$, as shown by Proposition 2.15. Then, the bounded fixed point of $F_{\mathcal{I}}$ generated by $F_{\mathcal{I}_{\mathsf{co}}}$ is constructed as follows:

- First, we take the least fixed point of $F_{\mathcal{I}} \cup F_{\mathcal{I}_{\mathsf{co}}}$, which plays the role of the bound for the next step.
- Then, we take the greatest (post-)fixed point of $F_{\mathcal{I}}$ below such a bound.

To make this correspondence precise, the key property is the following:

PROPOSITION 3.12 : Let $\mathcal{I}$ be an inference system on the universe $\mathcal{U}$ and consider $X \subseteq \mathcal{U}$, then $F_{(\mathcal{I}_{|X})} = (F_{\mathcal{I}})_{\sqcap X}$.

Proof: We have to show that, for $S \subseteq \mathcal{U}$, $(F_{\mathcal{I}})_{\sqcap X}(S) = F_{(\mathcal{I}_{|X})}(S)$.

If $j \in (F_{\mathcal{I}})_{\sqcap X}(S)$, then we have $j \in X$ and $j \in F_{\mathcal{I}}(S)$, hence there is $\langle Pr, j \rangle \in \mathcal{I}$ such that $Pr \subseteq S$; therefore, by definition of $\mathcal{I}_{|X}$, we get $\langle Pr, j \rangle \in \mathcal{I}_{|X}$, and this implies that $j \in F_{(\mathcal{I}_{|X})}(S)$.

Conversely, if $j \in F_{(\mathcal{I}_{|X})}(S)$, then there exists $\langle Pr, j \rangle \in \mathcal{I}_{|X}$ such that $Pr \subseteq S$. By definition of $\mathcal{I}_{|X}$, we have that $\langle Pr, j \rangle \in \mathcal{I}$ and $j \in X$, therefore $j \in F_{\mathcal{I}}(S)$ and $j \in X$, thus $j \in (F_{\mathcal{I}})_{\sqcap X}(S)$. □

Recall from Definition 3.3 and Corollary 3.4 that $\nu[F_{\mathcal{I}}, F_{\mathcal{I}_{\mathsf{co}}}] = \nu(F_{\mathcal{I}})_{\sqcap \mu(F_{\mathcal{I}} \cup F_{\mathcal{I}_{\mathsf{co}}})}$, namely, the greatest post-fixed point of $F_{\mathcal{I}}$ included in $\mu(F_{\mathcal{I}} \cup F_{\mathcal{I}_{\mathsf{co}}})$. Then, we have the following theorem:

THEOREM 3.13 : $\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] = \nu[F_{\mathcal{I}}, F_{\mathcal{I}_{\mathsf{co}}}]$.

Proof: It is easy to see that $F_{\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}} = F_{\mathcal{I}} \cup F_{\mathcal{I}_{\mathsf{co}}}$, hence, by Theorem 2.23, we get $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!] = \mu(F_{\mathcal{I}} \cup F_{\mathcal{I}_{\mathsf{co}}})$. Then, applying Theorem 2.24, Proposition 3.12 and Definitions 3.1 and 3.3, we get

$$\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] = \nu[\![\mathcal{I}_{|\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]}]\!] = \nu(F_{\mathcal{I}})_{\sqcap \mu(F_{\mathcal{I}} \cup F_{\mathcal{I}_{\mathsf{co}}}} = \nu[F_{\mathcal{I}}, F_{\mathcal{I}_{\mathsf{co}}}]$$

□

As an immediate consequence of Corollary 3.11 and Theorem 3.13, we get the following iterative characterisation of the interpretation of an inference system with corules.

COROLLARY 3.14 : If $F_{\mathcal{I}}$ is downward ω-continuous, then

$$\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] = \bigcap_{n \in \mathbf{N}} F^n_{\mathcal{I}}(\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!])$$

As already noticed, this iterative characterisation requires an additional condition on the inference operator, which can be ensured by conditions on rules (cf. Section 2.5).

We conclude this section by showing that the inductive and the coinductive interpretations of $\mathcal{I}$ are particular cases of the interpretation generated by corules, that is, they can be recovered by specific choices of corules. We denote by $\emptyset$ the empty inference system and by $\mathcal{I}_{\mathcal{U}}$ the inference system containing exactly one axiom for each judgement $j \in \mathcal{U}$. Then we have the following corollary:

COROLLARY 3.15 : Let $\mathcal{I}$ be an inference system on the universe $\mathcal{U}$, then the following hold:

1. $\mu[\![\mathcal{I}]\!] = \nu[\![\mathcal{I}, \emptyset]\!]$,
2. $\nu[\![\mathcal{I}]\!] = \nu[\![\mathcal{I}, \mathcal{I}_{\mathcal{U}}]\!]$.

Proof: The thesis follows from Theorem 3.13 and Proposition 3.8, noting that, for all $S \subseteq \mathcal{U}$, we have $F_{\emptyset}(S) = \emptyset = K_{\emptyset}(S)$ and $F_{\mathcal{I}_{\mathcal{U}}}(S) = \mathcal{U} = K_{\mathcal{U}}(S)$. □

Intuitively, when we construct the interpretation of $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$ (cf. Definition 3.2), if we have no corules, that is, $\mathcal{I}_{\mathsf{co}} = \emptyset$ (Item 1), then we keep only those rules of $\mathcal{I}$ whose conclusion belongs to $\mu[\![\mathcal{I}]\!]$, hence we can derive all and only judgements in $\mu[\![\mathcal{I}]\!]$; on the other hand, if we have one coaxiom for each judgement, that is, $\mathcal{I}_{\mathsf{co}} = \mathcal{I}_{\mathcal{U}}$ (Item 2), we do not remove any rule from $\mathcal{I}$ and so we get exactly its coinductive interpretation.

3.3 Proof trees for corules

In this section we formalize several proof-theoretic characterizations of the semantics of inference systems with corules (cf. Definition 3.2). To carry out the proof of equivalence, we rely on the fixed point characterisation of such semantics presented in Section 3.2.2.

In the following assume an inference system $\mathcal{I}$ on the universe $\mathcal{U}$ and recall that, given $X \subseteq \mathcal{U}$, $\mathcal{I}_{|X}$ is the subset of $\mathcal{I}$ consisting only of those rules with conclusion in X. The first proof-theoretic characterisation follows from the general framework presented in Section 2.3 for standard inference systems. Indeed, the following corollary follows immediately from Theorem 2.24 and Proposition 3.12.

COROLLARY 3.16 : Let $X \subseteq \mathcal{U}$, then $\nu[\![\mathcal{I}_{|X}]\!] = \mathsf{r}_!(\nu T_{(\mathcal{I}_{|X})}) = \nu((F_{\mathcal{I}})_{\sqcap X})$.

Then, to get the first proof-theoretic characterisation, we just have to describe proof trees in $\nu T_{\mathcal{I}_{|X}}$, which can be done as follows:

PROPOSITION 3.17 : Let $X \subseteq \mathcal{U}$, then $\tau \in \nu T_{\mathcal{I}_{|X}}$ iff τ is a proof tree in $\mathcal{I}$ and, for all $\alpha \in \mathsf{N}(\tau)$, $\tau(\alpha) \in X$.

Proof: Let $\tau \in \nu T_{\mathcal{I}_{|X}}$, then, by Lemma 2.21, τ is a proof tree in $\mathcal{I}_{|X}$, hence, by Definition 2.5, we have that, for all $\alpha \in \mathsf{N}(\tau)$, $\langle \mathsf{r}_!(\mathsf{chl}_\tau(\alpha)), \tau(\alpha)\rangle \in \mathcal{I}_{|X}$. By definition of $\mathcal{I}_{|X}$, we have $\mathcal{I}_{|X} \subseteq \mathcal{I}$, hence τ is a proof tree in $\mathcal{I}$, and $\tau(\alpha) \in X$, as needed. □

In other words, $\nu T_{\mathcal{I}_{|X}}$ is the set of the proof trees in $\mathcal{I}$ whose nodes all belong to X.

The first proof-theoretic characterisation of the interpretation of an inference system with corules $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle$ is a particular case of the above proposition:

COROLLARY 3.18 : Let $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle$ be an inference system with corules. Then the following are equivalent:

1. $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle \vdash_\nu j$
2. there exists a proof tree τ for j in $\mathcal{I}$ such that each node of τ has a well-founded proof tree in $\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}$

Proof: By Definition 3.2, we have $\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] = \nu[\![\mathcal{I}_{|\mu[\![\mathcal{I}\cup\mathcal{I}_{\mathsf{co}}]\!]}]\!]$, hence, by Corollary 3.16 and Proposition 3.17, we get that $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle \vdash_\nu j$ implies that there is a proof tree τ for j in $\mathcal{I}$ such that, for each node $\alpha \in \mathsf{N}(\tau)$, $\tau(\alpha) \in \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$. Therefore, by Theorem 2.23, we get that $\tau(\alpha)$ has a well-founded proof tree in $\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}$, as needed. □

APPROXIMATED PROOF TREES For the second proof-theoretic characterization, we need to define *approximated proof trees* in an inference system with corules.

DEFINITION 3.19 : Let $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle$ be an inference system with corules. For all $n \in \mathbb{N}$, the sets $\mathcal{T}^n_{\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle}$ of *approximated proof trees of level n in* $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle$ is defined by $\mathcal{T}^n_{\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle} = T^n_{\mathcal{I}}(\mu T_{\mathcal{I}\cup\mathcal{I}_{\mathsf{co}}})$.

It is easy to check that, for all $n \in \mathbb{N}$, $\mathcal{T}^n_{\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle} \subseteq \mu T_{\mathcal{I}\cup\mathcal{I}_{\mathsf{co}}}$, that is, relying on Lemma 2.20, approximated proof trees are well-founded proof trees in $\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}$. More precisely, an approximated proof tree is constructed starting from well-founded proof trees in $\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}$ applying *only* rules from $\mathcal{I}$. In other words, an approximated proof tree of level n in $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle$ is a well-founded proof tree in $\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}$ where corules can only be used at depth greater than or equal to n.

Another simple property of approximated proof trees is stated in the following proposition.

PROPOSITION 3.20 : If $\tau \in \mathcal{T}^n_{\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle}$, $\alpha \in \mathsf{N}(\tau)$ and $|\alpha| = k \leq n$, then $\tau_{|\alpha} \in \mathcal{T}^{n-k}_{\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle}$.

Proof: We proceed by induction on $|\alpha|$. If $|\alpha| = 0$, then $\alpha = \varepsilon$, hence $\tau_{|_\varepsilon} = \tau \in \mathcal{T}^n_{\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle}$. Assume $|\alpha| = k + 1$, hence $\alpha = \beta j$, with $\beta \in \mathtt{N}(\tau)$ and $|\beta| = k$. Therefore, by induction hypothesis, $\tau_{|_\beta} \in \mathcal{T}^{n-k}_{\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle}$ and, since $\tau_{|_\alpha} = (\tau_{|_\beta})_{|_j} \in \mathsf{dst}(\tau_{|_\beta})$ and $\mathcal{T}^{n-k}_{\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle} = T_{\mathcal{I}}(\mathcal{T}^{n-k-1}_{\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle})$, by definition of $T_{\mathcal{I}}$, we get $\tau_{|_\alpha} \in \mathsf{dst}(\tau_{|_\beta}) \subseteq \mathcal{T}^{n-k-1}_{\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle}$, as needed. □

Relying on the relationship between $T_{\mathcal{I}}$ and $F_{\mathcal{I}}$ (cf. Proposition 2.16) and on the equivalence between proof-theoretic and model-theoretic semantics in the inductive case (cf. Theorem 2.23), we can derive the following theorem, providing an equivalent model-theoretic characterisation of judgements derivable by approximated proof trees.

THEOREM 3.21 : Let $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$ be an inference system with corules. For all $n \in \mathbb{N}$, the following are equivalent:

1. $j \in F^n_{\mathcal{I}}(\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!])$
2. j has an approximated proof tree of level n in $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$

that is, $\mathsf{r}_!(\mathcal{T}^n_{\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle}) = F^n_{\mathcal{I}}(\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!])$.

Proof: By Definition 3.19 and Proposition 2.16, we have $\mathsf{r}_!(\mathcal{T}^n_{\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle}) = \mathsf{r}_!(T^n_{\mathcal{I}}(\mu T_{\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}})) = F^n_{\mathcal{I}}(\mathsf{r}_!(\mu T_{\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}})$; then, by Theorem 2.23, we get the thesis. □

Then, we immediately get the second proof-theoretic characterisation:

COROLLARY 3.22 : Let $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$ be an inference system with corules. The following are equivalent:

1. $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle \vdash_\nu j$
2. there exists a proof tree τ for j in $\mathcal{I}$ such that each node of τ has an approximated proof tree of level n in $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$, for all $n \in \mathbb{N}$.

Proof: By Theorem 3.13, Proposition 3.9 (2), Corollary 3.16 and Proposition 3.17, we get that, $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle \vdash_\nu j$ iff there exists a proof tree τ for j in $\mathcal{I}$ such that, for each node $\alpha \in \mathtt{N}(\tau)$, $\tau(\alpha) \in \bigcap_{n \in \mathbb{N}} F^n_{\mathcal{I}}(\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!])$, that is, for all $n \in \mathbb{N}$, $\tau(\alpha) \in F^n_{\mathcal{I}}(\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!])$, hence, by Theorem 3.21, we get the thesis. □

If the hypotheses of Corollary 3.11 are satisfied, namely, if the inference operator $F_{\mathcal{I}}$ is downward ω-continuous, then we get a simpler equivalent proof-theoretic characterization.

COROLLARY 3.23 : Let $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$ be an inference system with corules where $F_{\mathcal{I}}$ is downward ω-continuous. The following are equivalent:

1. $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle \vdash_\nu j$
2. j has an approximated proof tree of level n in $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$, for all $n \in \mathbb{N}$.

Proof: By Corollary 3.11, we have $\nu[\![\mathcal{I}, \mathcal{I}_{\text{co}}]\!] = \bigcap_{n\in\mathbb{N}} F_{\mathcal{I}}(F_{\mathcal{I}\cup\mathcal{I}_{\text{co}}})$, then the thesis follows by Theorem 3.21. □

APPROXIMATING PROOF SEQUENCE In order to define the last proof-theoretic characterization (Theorem 3.26), we need to introduce a richer structure on trees. For basic definitions about trees we refer to Section 2.1.1. Assume a set A and consider the set of trees on A, denoted by $\mathcal{T}_A$. We define a family of equivalence relations on $\mathcal{T}_A$, as detailed below. For all $\tau \in \mathcal{T}_A$, we denote by $\lfloor\tau\rfloor_n$ the *n-truncation of* τ, which is the tree $\langle \mathsf{r}(\tau), \mathsf{N}_n(\tau)\rangle$, where $\mathsf{N}_n(\tau)$ is the subset of nodes of τ at depth less than or equal to n, that is, $\mathsf{N}_n(\tau) = \{\alpha \in \mathsf{N}(\tau) \mid |\alpha| \leq n\}$. Therefore, $\lfloor\tau\rfloor_n$ is the tree obtained by cutting τ at depth n. It is easy to see that $\lfloor\lfloor\tau\rfloor_n\rfloor_k = \lfloor\tau\rfloor_{\min\{n,k\}}$ and $\mathsf{N}(\tau) = \bigcup_{n\in\mathbb{N}} \mathsf{N}_n(\tau)$.

For all $n \in \mathbb{N}$, we define the relation $\approx_n$ on $\mathcal{T}_A$ as follows:

$$\tau \approx_n \tau' \text{ iff } \lfloor\tau\rfloor_n = \lfloor\tau'\rfloor_n$$

Intuitively, these equivalence relations model equality of trees up to a fixed depth, that is, if $\tau \approx_n \tau'$, then τ and τ' share the same root and the same nodes up to depth n. In other words, they approximate equality, that is, we have $\tau = \tau'$ iff, for all $n \in \mathbb{N}$, $\tau \approx_n \tau'$.

We now focus on sequences $(\tau_n)_{n\in\mathbb{N}}$ of trees related by $\approx_n$, for increasing n, that is, for all $n \in \mathbb{N}$, $\tau_n \approx_n \tau_{n+1}$. When n grows, trees in the sequence share a larger and larger portion of nodes, hence, in a sense, they tend to a limit tree. This intuition is formalised by the following theorem.

THEOREM 3.24 : Let $(\tau_n)_{n\in\mathbb{N}}$ be a sequence of trees, such that, for all $n \in \mathbb{N}$, $\tau_n \approx_n \tau_{n+1}$. Then, there exists a unique tree τ such that, for all $n \in \mathbb{N}$, $\tau_n \approx_n \tau$.

Proof: First of all, note that, for all $n \in \mathbb{N}$, $\mathsf{r}(\tau_0) = \mathsf{r}(\tau_n)$. Then, let $n \in \mathbb{N}$, and prove by induction on k that, for all $k \geq n$, $\tau_n \approx_n \tau_k$. If $k = n$, this is immediate. If $k > n$, then, by induction hypothesis, $\tau_n \approx_n \tau_{k-1}$, hence we have to show that $\tau_{k-1} \approx_n \tau_k$. Since $\tau_{k-1} \approx_{k-1} \tau_k$, we have $\lfloor\tau_{k-1}\rfloor_{k-1} = \lfloor\tau_k\rfloor_{k-1}$. Then, since $n \leq k-1$, we get $\lfloor\tau_{k-1}\rfloor_n = \lfloor\lfloor\tau_{k-1}\rfloor_{k-1}\rfloor_n = \lfloor\lfloor\tau_k\rfloor_{k-1}\rfloor_n = \lfloor\tau_k\rfloor_n$, that is, $\tau_{k-1} \approx_n \tau_k$, as needed.

Let τ be the tree defined by $\mathsf{r}(\tau) = \mathsf{r}(\tau_0)$ and $\mathsf{N}(\tau) = \bigcup_{n\in\mathbb{N}} \mathsf{N}_n(\tau_n)$. For all $n \in \mathbb{N}$, we know that $\mathsf{r}(\tau) = \mathsf{r}(\tau_0) = \mathsf{r}(\tau_n)$ and $\mathsf{N}_n(\tau_n) \subseteq \mathsf{N}_n(\tau)$, by definition, hence, to conclude, we have only to check the other inclusion. If $\alpha \in \mathsf{N}_n(\tau)$, then $|\alpha| \leq n$ and $\alpha \in \mathsf{N}_k(\tau_k)$, hence $|\alpha| \leq k$ and, if $h = \min\{n, k\}$, $\alpha \in \mathsf{N}_h(\tau_k)$. From what we observed above, we get $\tau_n \approx_h \tau_k$, hence $\alpha \in \mathsf{N}_h(\tau_n) \subseteq \mathsf{N}_n(\tau_n)$, as needed.

Finally, to show uniqueness, let τ' be a tree such that, for all $n \in \mathbb{N}$, $\tau' \approx_n \tau_n$, hence we get $\lfloor\tau'\rfloor_n = \lfloor\tau_n\rfloor_n = \lfloor\tau\rfloor_n$, that is, $\tau' \approx_n \tau$, for all $n \in \mathbb{N}$, which implies $\tau' = \tau$, as needed. □

Given a sequence $(\tau_n)_{n\in\mathbb{N}}$ such that, for all $n \in \mathbb{N}$, $\tau_n \approx_n \tau_{n+1}$, we denote by $\bigsqcup_{n\in\mathbb{N}} \tau_n$ the unique τ constructed by Theorem 3.24.

It is well known that trees carry a complete metric space structure (Arnold and Nivat, 1980; Courcelle, 1983) and, even if our notion of tree is slightly different from that adopted in these works, we can recover the same metric on our trees, using the equivalence relations introduced earlier. The metric is defined as follows:

$$d(\tau, \tau') = 2^{-h} \qquad h = \inf\{n \in \mathbb{N} \mid \tau \not\approx_n \tau'\}$$

with $\inf \emptyset = \infty$ and $2^{-\infty} = 0$. It is easy to see that a sequence $(\tau_n)_{n\in\mathbb{N}}$ such that $\tau_n \approx_n \tau_{n+1}$, like that considered in Theorem 3.24, is a Cauchy sequence in the metric space; indeed, $d(\tau_n, \tau_{n+1}) \leq 2^{-n}$. Therefore, such sequences converge also in the metric space, and the limit is the same. A deeper comparison between this relation and the standard metric structure on trees will be matter of further work.

We can now introduce the concept that will allow the last proof-theoretic characterization.

DEFINITION 3.25 (Approximating proof sequence): Let $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$ be an inference system with corules and $j \in \mathcal{U}$ a judgement. An *approximating proof sequence* for j is a sequence of proof trees $(\tau_n)_{n\in\mathbb{N}}$ for j such that $\tau_n \in \mathcal{T}^n_{\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle}$ and $\tau_n \approx_n \tau_{n+1}$, for all $n \in \mathbb{N}$.

Note also that all trees in an approximating proof sequence are well-founded proof trees in $\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}$. Intuitively, this notion represents the growth of a proof for j in $\mathcal{I}$ approximated using corules. We now prove our last theorem, characterizing $\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$ in terms of approximating proof sequences.

THEOREM 3.26 : Let $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$ be an inference system with corules. The following are equivalent

1. $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle \vdash_\nu j$
2. j has an approximating proof sequence $(t_n)_{n\in\mathbb{N}}$

Proof: We prove 1 implies 2. We define trees $\tau_{j,n}$ for $j \in \nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$ and $n \in \mathbb{N}$ by induction on n, as detailed below. By Corollary 3.18, we know that every judgement $j \in \nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$ has a well-founded proof tree in $\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}$, that is, a proof tree in $\mathcal{T}^0_{\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle}$ rooted in j: we select one of these trees and call it $\tau_{j,0}$. Furthermore, since $\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$ is a post-fixed point, for any $j \in \nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$ we can select a rule $\langle Pr_j, j \rangle \in \mathcal{I}$ with $Pr_j \subseteq \nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$; hence $\tau_{j,n+1}$ can be defined as follows:

$$\tau_{j,n+1} = \frac{\{\tau_{j',n} \mid j' \in Pr_j\}}{j}$$

Clearly, by construction, for all $j \in \nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$ and for all $n \in \mathbb{N}$, $\mathsf{r}(\tau_{j,n}) = j$ and $\tau_{j,n} \in \mathcal{T}^n_{\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle}$. We show by induction on n that, for all $n \in \mathbb{N}$ and for all $j \in \nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$, $\tau_{j,n} \approx_n \tau_{j,n+1}$.

Case: $n = 0$ By construction, we have $\mathsf{r}(\tau_{j,0}) = j = \mathsf{r}(\tau_{j,1})$ and $\mathsf{N}_0(\tau_{j,0}) = \{\varepsilon\} = \mathsf{N}_0(\tau_{j,1})$, hence $\tau_{j,0} \approx_0 \tau_{j,1}$, as needed.

Case: $n > 0$ We assume the thesis for $n-1$ and prove it for n, hence we have to show that $\tau_{j,n} \approx_n \tau_{j,n+1}$. By construction, we have

$$\tau_{j,n} = \frac{\{\tau_{j',n-1} \mid j' \in Pr_j\}}{j} \qquad \tau_{j,n+1} = \frac{\{\tau_{j',n} \mid j' \in Pr_j\}}{j}$$

By induction hypothesis, we know that $\tau_{j',n-1} \approx_{n-1} \tau_{j',n}$ for all $j' \in Pr_j$. Therefore we have

$$\begin{aligned}\mathsf{N}_n(\tau_{j,n}) &= \{\varepsilon\} \cup \bigcup_{j' \in Pr_j} j'\mathsf{N}_{n-1}(\tau_{j',n-1}) \\ &= \{\varepsilon\} \cup \bigcup_{j' \in Pr_j} j'\mathsf{N}_{n-1}(\tau_{j',n}) \\ &= \mathsf{N}_n(\tau_{j,n+1})\end{aligned}$$

Therefore, since $\mathsf{r}(\tau_{j,n}) = j = \mathsf{r}(\tau_{j,n+1})$, we have the thesis.

We prove 2 implies 1. By hypothesis, j has an approximating proof sequence $(\tau_n)_{n\in\mathbb{N}}$. We set $\tau = \bigsqcup_{n\in\mathbb{N}} \tau_n$ and prove that τ is a proof tree in $\mathcal{I}$ for j. By Theorem 3.24, we have that $\tau_n \approx_n \tau$, for all $n \in \mathbb{N}$, hence, we get $j = \mathsf{r}(\tau_0) = \mathsf{r}(\tau)$. Consider $\alpha \in \mathsf{N}(()\tau)$ and set $n = |\alpha| + 1$. By construction of τ, we have that $\alpha \in \mathsf{N}_n(\tau_n)$ and, for all $j' \in \mathcal{U}$, $\alpha j' \in \mathsf{N}(\tau)$ iff $\alpha j' \in \mathsf{N}_n(\tau_n)$, since $|\alpha j'| = n$, hence, $\mathsf{r}_!(\mathsf{chl}_\tau(\alpha)) = \mathsf{r}_!(\mathsf{chl}_{\tau_n}(\alpha))$, as $\tau_n \approx_n \tau$. Since $\tau_n \in \mathcal{T}^n_{\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle}$, by Proposition 3.20, we get $(\tau_n)_{|_\alpha} \in \mathcal{T}^1_{\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle}$, hence, by Definition 3.19, we have $\langle \mathsf{r}_!(\mathsf{chl}_{(\tau_n)_{|_\alpha}}(\varepsilon)), (\tau_n)_{|_\alpha}(\varepsilon)\rangle \in \mathcal{I}$. Therefore, since $\mathsf{chl}_{(\tau_n)_{|_\alpha}}(\varepsilon) = \mathsf{chl}_{\tau_n}(\alpha) = \mathsf{chl}_\tau(\alpha)$ and $(\tau_n)_{|_\alpha}(\varepsilon) = \tau_n(\alpha) = \tau(\alpha)$, we get $\langle \mathsf{r}_!(\mathsf{chl}_\tau(\alpha)), \tau(\alpha)\rangle \in \mathcal{I}$, and this proves τ is a proof tree. □

3.4 Reasoning with corules

In this section we discuss proof techniques for inference systems with corules.

Let $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle$ be an inference system with corules on the universe $\mathcal{U}$. As discussed in Section 2.4, we are typically interested in comparing the interpretation of $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle$ to a set of judgements $\mathcal{S} \subseteq \mathcal{U}$ (*specification*), focusing on *soundness* ($\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] \subseteq \mathcal{S}$) and *completeness* ($\mathcal{S} \subseteq \nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$) properties. Proving both properties amounts to say that the inference system with corules actually defines the given specification $\mathcal{S}$.

COMPLETENESS PROOFS To show completeness, we rely on the fixed point characterisation of $\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$ (cf. Theorem 3.13), using the principle $(\nu_{\mathrm{B}}\,\mathrm{P})$ (cf. p.38) associated with the bounded fixed point. We rephrase $(\nu_{\mathrm{B}}\,\mathrm{P})$ (cf. p.38) in the specific case of inference systems with corules as follows:

PROPOSITION 3.27 (Bounded coinduction principle): Let $\mathcal{S} \subseteq \mathcal{U}$ be a set of judgements. If the following hold

BOUNDEDNESS $\mathcal{S} \subseteq \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$ and

CONSISTENCY $\mathcal{S} \subseteq F_{\mathcal{I}}(\mathcal{S})$

then $\mathcal{S} \subseteq \nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$.

We call this principle the *bounded coinduction principle*.

EXAMPLE 3.28 : Let us illustrate the technique on the inference system with corules $\langle \mathcal{I}^{>0}, \mathcal{I}_{\mathsf{co}}^{>0} \rangle$ which defines the judgement $\mathsf{allPos}_{\mathbb{B}}(l, b)$. We report here the definition from Section 3.1, for the reader's convenience.

$$\text{(A-E)}\ \frac{}{\mathsf{allPos}_{\mathbb{B}}(\varepsilon, \mathrm{T})} \qquad \text{(A-F)}\ \frac{}{\mathsf{allPos}_{\mathbb{B}}(x{:}l, \mathrm{F})}\ x \leq 0$$

$$\text{(A-T)}\ \frac{\mathsf{allPos}_{\mathbb{B}}(l, b)}{\mathsf{allPos}_{\mathbb{B}}(x{:}l, b)}\ x > 0 \qquad \text{(CO-A)}\ \frac{}{\mathsf{allPos}_{\mathbb{B}}(l, \mathrm{T})}$$

Recall that a possibly infinite list of integers is said positive if it contains only positive elments. Let us define the set of judgements $\mathcal{S}^{>0}$ as follows: $\mathsf{allPos}_{\mathbb{B}}(l, b) \in \mathcal{S}^{>0}$ iff l is positive and $b = \mathrm{T}$ or l is not positive and $b = \mathrm{F}$. Completeness, $\mathcal{S}^{>0} \subseteq \nu[\![\mathcal{I}^{>0}, \mathcal{I}_{\mathsf{co}}^{>0}]\!]$, can be stated as follows:

> If l is positive, then $\langle \mathcal{I}^{>0}, \mathcal{I}_{\mathsf{co}}^{>0} \rangle \vdash_{\nu} \mathsf{allPos}_{\mathbb{B}}(l, \mathrm{T})$,
> otherwise $\langle \mathcal{I}^{>0}, \mathcal{I}_{\mathsf{co}}^{>0} \rangle \vdash_{\nu} \mathsf{allPos}_{\mathbb{B}}(l, \mathrm{F})$.

The proof is by bounded coinduction, hence we have to prove the following:

BOUNDEDNESS if $\mathsf{allPos}_{\mathbb{B}}(l, b) \in \mathcal{S}^{>0}$, then $\mathcal{I}^{>0} \cup \mathcal{I}_{\mathsf{co}}^{>0} \vdash_{\mu} \mathsf{allPos}_{\mathbb{B}}(l, b)$,

CONSISTENCY if $\mathsf{allPos}_{\mathbb{B}}(l, b) \in \mathcal{S}^{>0}$, then there is a rule $\langle Pr, \mathsf{allPos}_{\mathbb{B}}(l, b) \rangle \in \mathcal{I}^{>0}$ such that $Pr \subseteq \mathcal{S}^{>0}$.

To prove boundedness, we have to show that, if $\mathsf{allPos}_{\mathbb{B}}(l, b) \in \mathcal{S}^{>0}$, then $\mathsf{allPos}_{\mathbb{B}}(l, b)$ has a *finite* proof tree in $\mathcal{I}^{>0} \cup \mathcal{I}_{\mathsf{co}}^{>0}$. This can be easily done, as follows. If l is positive, then $b = \mathrm{T}$ and $\mathsf{allPos}_{\mathbb{B}}(l, \mathrm{T})$ is derived by (CO-A). Otherwise, $b = \mathrm{F}$ and $l = x_0{:}\ldots{:}x_n{:}l'$ with $x_n \leq 0$ and $x_i > 0$, for all $i \in 0..n-1$, hence we can reason by arithmetic induction on n. Indeed, for $n = 0$, $\mathsf{allPos}_{\mathbb{B}}(l, \mathrm{F})$ is derived by (A-F), and, for $n > 0$, it is derived by (A-T) where the premise is derivable by induction hypothesis.

To prove consistency, we proceed as follows. Assume $\mathsf{allPos}_{\mathbb{B}}(l, b) \in \mathcal{S}^{>0}$ and distinguish the following cases.

Case: $l = \varepsilon$ We have $b = \mathrm{T}$ and so the thesis follows by (A-E).

Case: $l = x{:}l'$ *and* $x \leq 0$ We have $b = \mathrm{F}$ and so the thesis follows by (A-F).

Case: $l = x{:}l'$ *and* $x > 0$ We have that l is positive iff l' is positive, hence $\mathsf{allPos}_{\mathbb{B}}(l', b) \in \mathcal{S}^{>0}$ and so the thesis follows by (A-T).

SOUNDNESS PROOFS To show soundness, we exploit the refinement of the definition of the interpretation of $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$ provided in Proposition 3.9, which gives us the inclusion $\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] \subseteq \bigcap_{n \in \mathbb{N}} F_{\mathcal{I}}^n(\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!])$. Hence, to prove soundness, that is, $\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] \subseteq \mathcal{S}$, we have to show that $\bigcap_{n \in \mathbb{N}} F_{\mathcal{I}}^n(\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]) \subseteq \mathcal{S}$. This can be done in two ways:

- either we reason by contraposition, proving that judgements not in $\mathcal{S}$ do not belong to $F_{\mathcal{I}}^n(\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!])$, for some $n \in \mathbb{N}$,

- or we find a sequence of sets $(\mathcal{S}_n)_{n\in\mathbb{N}}$ such that $\bigcap_{n\in\mathbb{N}} \mathcal{S}_n \subseteq \mathcal{S}$, and prove that $F^n_{\mathcal{I}}(\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]) \subseteq \mathcal{S}_n$, for all $n \in \mathbb{N}$.

The advantage of the second approach is that it can be proved by arithmetic induction on n.

By the proof-theoretic characterisation in terms of approximated proof trees given in Corollary 3.22, we can rephrase the above techniques as follows:

- either we reason by contraposition, proving that judgements not in $\mathcal{S}$ fail to have an approximated proof tree of level n, for some $n \in \mathbb{N}$,
- or we find a sequence of sets $(\mathcal{S}_n)_{n\in\mathbb{N}}$ such that $\bigcap_{n\in\mathbb{N}} \mathcal{S}_n \subseteq \mathcal{S}$, and prove that, for all $n \in \mathbb{N}$, if a judgement has an approximated proof tree of level n, then it belongs to $\mathcal{S}_n$.

EXAMPLE 3.29 : We illustrate the technique again on the example $\langle \mathcal{I}^{>0}, \mathcal{I}^{>0}_{\mathsf{co}} \rangle$. The soundness statement can be expressed as follows:

> if $\langle \mathcal{I}^{>0}, \mathcal{I}^{>0}_{\mathsf{co}} \rangle \vdash_\nu \mathsf{allPos}_{\mathbb{B}}(l, b)$, then,
> if $b = \mathrm{T}$, l is positive, and if $b = \mathrm{F}$, l is not positive.

Given a possibly infinite list l, let us denote by $|l| \in \mathbb{N} \cup \{\infty\}$ the length of l and, for each $n < |l|$, by $l(n)$ the n-th element of l. For all $n \in \mathbb{N}$, define $\mathcal{S}^{>0}_n$ as follows: $\mathsf{allPos}_{\mathbb{B}}(l, \mathrm{T}) \in \mathcal{S}^{>0}_n$ iff, for all $k < \min\{n, |l|\}$, $l(k) > 0$, and $\mathsf{allPos}_{\mathbb{B}}(l, \mathrm{F}) \in \mathcal{S}^{>0}_n$ iff l is not positive. It is easy to see that $\bigcap_{n\in\mathbb{N}} \mathcal{S}^{>0}_n \subseteq \mathcal{S}^{>0}$, because l is positive iff all its finite prefixes are positive. Assume $\langle \mathcal{I}^{>0}, \mathcal{I}^{>0}_{\mathsf{co}} \rangle \vdash_\nu \mathsf{allPos}_{\mathbb{B}}(l, b)$. We can easily prove by induction on rules in $\mathcal{I}^{>0} \cup \mathcal{I}^{>0}_{\mathsf{co}}$ that, if $\mathcal{I}^{>0} \cup \mathcal{I}^{>0}_{\mathsf{co}} \vdash_\mu \mathsf{allPos}_{\mathbb{B}}(l, \mathrm{F})$, then l is not positive. There are only two relevant cases: for rule (A-F), we have $l = x{:}l'$ with $x \leq 0$, hence l is not positive, and for rule (A-T), we have $l = x{:}l'$ and l' is not positive by induction hypothesis, hence so is l. This implies that $\mathsf{allPos}_{\mathbb{B}}(l, \mathrm{F}) \in \mathcal{S}^{>0}_n$, for all $n \in \mathbb{N}$, and, since $F^n_{\mathcal{I}^{>0}}(\mu[\![\mathcal{I}^{>0} \cup \mathcal{I}^{>0}_{\mathsf{co}}]\!]) \subseteq \mu[\![\mathcal{I}^{>0} \cup \mathcal{I}^{>0}_{\mathsf{co}}]\!]$, we have the thesis.

Now, for $b = \mathrm{T}$, we prove by arithmetic induction on n that, if $\mathsf{allPos}_{\mathbb{B}}(l, \mathrm{T}) \in F^n_{\mathcal{I}^{>0}}(\mu[\![\mathcal{I}^{>0} \cup \mathcal{I}^{>0}_{\mathsf{co}}]\!])$, then, for all $k < \min\{n, |l|\}$, $l(k) > 0$. If $n = 0$, then there is nothing to prove, as $\min\{0, |l|\} = 0$ and there is no $k < 0$. If $n > 0$, then there is a rule in $\mathcal{I}^{>0}$ with conclusion $\mathsf{allPos}_{\mathbb{B}}(l, \mathrm{T})$ whose premises are in $F^{n-1}_{\mathcal{I}^{>0}}(\mu[\![\mathcal{I}^{>0} \cup \mathcal{I}^{>0}_{\mathsf{co}}]\!])$. We split cases on such rule.

Case: (A-E) We have $l = \varepsilon$, hence the thesis is trivial.

Case: (A-T) We have $l = x{:}l'$ with $x > 0$ and, by induction hypothesis, we know that, for all $k < \min\{n-1, |l'|\}$, $l'(k) > 0$. Then, if $k < \min\{n, |l|\} = 1 + \min\{n-1, |l'|\}$, we have two cases: if $k = 0$, then $l(k) = x > 0$, and, if $k > 0$, then $l(k) = l'(k-1)$, which is positive by the induction hypothesis.

3.5 Taming corules: advanced examples

In this section we will present some more examples of situations where corules can help to define judgements. These more involved examples will serve to

explain how to use corules, which kind of problems they can cope with, and how complex the interaction between corules and standard rules can be. In these examples, we use only corules with empty set of premises, namely, coaxioms, more examples going beyond coaxioms can be found in next chapters, see *e.g.*, Section 6.4.

3.5.1 A numerical example

It is well-known that real numbers in the closed interval $[0, 1]$ can be represented by infinite sequences $(d_i)_{i\in\mathbb{N}_{>0}}$ of digits, given some basis $b \geq 2$, that is, $d_i \in 0..b-1$, for all $i \in \mathbb{N}_{>0}$, where $\mathbb{N}_{>0}$ denotes the set of all positive natural numbers. Indeed, $(d_i)_{i\in\mathbb{N}_{>0}}$ represents the real number which is the limit of the series $\sum_{i=1}^{\infty} b^{-i}d_i$ in the standard complete metric space of real numbers with Euclidean distance (such a limit always exists by completeness, because the associated sequence of partial sums is always a Cauchy sequence). Such a representation is not unique for all rational numbers in $[0, 1]$ (except for the bounds 0 and 1) that can be represented by a finite sequence of digits followed by an infinite sequence of 0s; for instance, with $b = 10$, 0.42 can be represented either by the sequence $42\overline{0}$, or by the sequence $41\overline{9}$, where $\overline{d}$ denotes the infinite sequence containing just the digit d.

For brevity, for $r = (d_i)_{i\in\mathbb{N}_{>0}}$, $[\![r]\!]$ denotes $\sum_{i=1}^{\infty} b^{-i}d_i$ (that is, the real number represented by r). We want to define the judgement $\mathsf{add}(r_1, r_2, r, c)$ which holds iff $[\![r_1]\!] + [\![r_2]\!] = [\![r]\!] + c$ with c an integer number; that is, $\mathsf{add}(r_1, r_2, r, c)$ holds iff the addition of the two real numbers represented by the sequences r_1 and r_2 yields the real number represented by the sequence r with carry c. We will soon discover that, to get a complete definition for add, c is required to range over a proper superset of the set $\{0, 1\}$, differently from what one could initially expect.

We define the predicate add by the inference system with corules $\langle \mathcal{I}^{\mathsf{add}}, \mathcal{I}^{\mathsf{add}}_{\mathsf{co}} \rangle$ described below. Since we represent a real number in $[0, 1]$ by an infinite sequence of digits, we can always decompose r as $d{:}r$, where d is the first digit (corresponding to the exponent -1), and r is the rest of the sequence of digits. Hence, in the definition below, r, r_1, r_2 range over infinite sequences of digits, d_1, d_2 range over digits (between 0 and $b-1$), c is an integer and $\div$ and mod denote the integer division, and the remainder operator, respectively.

$$(\textsc{add})\ \frac{\mathsf{add}(r_1, r_2, r, c)}{\mathsf{add}(d_1{:}r_1, d_2{:}r_2, (s \bmod b){:}r, s \div b)} \quad s = d_1 + d_2 + c$$

$$(\textsc{co-add})\ \frac{}{\mathsf{add}(r_1, r_2, r, c)} \quad c \in \{-1, 0, 1, 2\}$$

As clearly emerges from the proof of completeness provided below, besides the obvious values 0 and 1, the values -1 and 2 have to be considered for the carry to ensure a complete definition of add because both $\mathsf{add}(\overline{0}, \overline{0}, \overline{9}, -1)$ and $\mathsf{add}(\overline{9}, \overline{9}, \overline{0}, 2)$ hold, and, hence, should be derivable; these two judgements allow the derivation of an infinite number of other correct judgements, as, for

instance, $\mathsf{add}(1\overline{0}, 1\overline{0}, 1\overline{9}, 0)$ and $\mathsf{add}(1\overline{9}, 1\overline{9}, 4\overline{0}, 0)$, respectively, as shown by the following infinite derivations:

$$\dfrac{\dfrac{\dfrac{\vdots}{\mathsf{add}(\overline{0},\overline{0},\overline{9},-1)}}{\mathsf{add}(\overline{0},\overline{0},\overline{9},-1)}}{\mathsf{add}(1\overline{0},1\overline{0},1\overline{9},0)} \qquad \dfrac{\dfrac{\dfrac{\vdots}{\mathsf{add}(\overline{9},\overline{9},\overline{0},2)}}{\mathsf{add}(\overline{9},\overline{9},\overline{0},2)}}{\mathsf{add}(1\overline{9},1\overline{9},4\overline{0},0)}$$

We sketch a proof of correctness: for all infinite sequences of digits r_1, r_2 and r, and all $c \in \{-1, 0, 1, 2\}$, $\langle \mathcal{I}^{\mathsf{add}}, \mathcal{I}^{\mathsf{add}}_{\mathsf{co}} \rangle \vdash_\nu \mathsf{add}(r_1, r_2, r, c)$ holds iff $[\![r_1]\!] + [\![r_2]\!] = [\![r]\!] + c$.

COMPLETENESS The completeness statement is as follows:

$$\text{if } [\![r_1]\!] + [\![r_2]\!] = [\![r]\!] + c, \text{ then } \langle \mathcal{I}^{\mathsf{add}}, \mathcal{I}^{\mathsf{add}}_{\mathsf{co}} \rangle \vdash_\nu \mathsf{add}(r_1, r_2, r, c)$$

The proof is by bounded coinduction (cf. Proposition 3.27).

By (CO-ADD) we trivially have that each judgement of shape $\mathsf{add}(r_1, r_2, r, c)$ such that $[\![r_1]\!] + [\![r_2]\!] = [\![r]\!] + c$ belongs to $\mu[\![\mathcal{I}^{\mathsf{add}} \cup \mathcal{I}^{\mathsf{add}}_{\mathsf{co}}]\!]$, because $c = [\![r_1]\!] + [\![r_2]\!] - [\![r]\!]$ and so $-1 \leq c \leq 2$, as $0 \leq [\![r']\!] \leq 1$ for all sequence r'.

To show consistency, let us assume that $[\![r'_1]\!] + [\![r'_2]\!] = [\![r']\!] + c'$ with $r'_1 = d_1{:}r_1$, $r'_2 = d_2{:}r_2$, $r' = d{:}r$. Let us set $s = b \cdot c' + d$, and $c = s - d_1 - d_2$, then $s \bmod b = d$ and $s \div b = c'$ because $d < b$, hence $\mathsf{add}(r'_1, r'_2, r', c')$ is the conclusion of (ADD) and, to conclude, we have to show that $[\![r_1]\!] + [\![r_2]\!] = [\![r]\!] + c$.

We first observe that by the properties of limits with respect to the usual arithmetic operations, and by definition of $[\![-]\!]$, for all infinite sequences r of digits, if $r = d{:}r'$, then $[\![r]\!] = b^{-1}(d + [\![r']\!])$; then, from the hypotheses we get the equality $d_1 + [\![r_1]\!] + d_2 + [\![r_2]\!] = d + [\![r]\!] + b \cdot c'$, hence $d_1 + [\![r_1]\!] + d_2 + [\![r_2]\!] = [\![r]\!] + s$, and, therefore, $[\![r_1]\!] + [\![r_2]\!] = [\![r]\!] + c$, as needed.

SOUNDNESS The soundness statement is as follows:

$$\text{if } \langle \mathcal{I}^{\mathsf{add}}, \mathcal{I}^{\mathsf{add}}_{\mathsf{co}} \rangle \vdash_\nu \mathsf{add}(r'_1, r'_2, r', c'), \text{ then } [\![r'_1]\!] + [\![r'_2]\!] = [\![r']\!] + c'$$

Given an infinite sequence of digits $r = (d_i)_{i \in \mathbb{N}_{>0}}$, we denote by $r[n]$ the sequence $d_1 \dots d_n$, with $n \geq 0$ ($r[0]$ is the empty sequence), and we write $\sum r[n]$ for $\sum_{i=1}^{n} b^{-1} \cdot d_i$ ($\sum r[0] = 0$), hence $[\![r]\!] = \lim_{n \to \infty}(\sum r[n])$. Let $r_1 = (d_{1,i})_{i \in \mathbb{N}_{>0}}$, $r_2 = (d_{2,i})_{i \in \mathbb{N}_{>0}}$ and $r = (d_i)_{i \in \mathbb{N}_{>0}}$ be infinite sequences of digits, hence $r_1 = d_{1,1}{:}r'_1$, $r_2 = d_{2,1}{:}r'_2$ and $r = d_1{:}r'$. It is easy to see that, if, for all $n \in \mathbb{N}$, $\sum r_1[n] + \sum r_2[n] = \sum r[n] + c - c_n \cdot b^{-n}$, for some $c_n \in \{-1, 0, 1, 2\}$, then $[\![r_1]\!] + [\![r_2]\!] = [\![r]\!] + c$; indeed, we have

$$[\![r_1]\!] + [\![r_2]\!] - [\![r]\!] - c = \lim_{n \to \infty} \left(\sum r_1[n] + \sum r_2[n] - \sum r[n] - c\right) = \lim_{n \to \infty} -c_n \cdot b^{-1} = 0$$

because c_n is limited, while b^{-n} decreases exponentially to 0.

To prove soundness, then we just have to show that, for all $n \in \mathbb{N}$, if $\mathsf{add}(r_1, r_2, r, c)$ has an approximated proof tree of level n, then $\sum r_1[n] + \sum r_2[n] = \sum r[n] + c - b^{-n} \cdot c_n$, for some $c_n \in \{-1, 0, 1, 2\}$. The proof is by induction on n.

Case: 0 It is easy to prove by induction on the derivation in $\mathcal{T}^0_{\langle \mathcal{I}^{\mathsf{add}}, \mathcal{I}^{\mathsf{add}}_{\mathsf{co}} \rangle}$ that $c \in \{-1, 0, 1, 2\}$. Then, the thesis is trivial taking $c_n = c$, because $\sum r_1[0] + \sum r_2[0] = 0 = \sum r[0] + c - c \cdot b^0$.

Case: $n + 1$ The judgement is derived by rule (ADD), hence $add(r'_1, r'_2, r', c')$ has an approximated proof tree of level n, and $d_1 = s \bmod b$, $c = s \div b$, and $s = d_{1,1} + d_{2,1} + c'$, that is, $d_{1,1} + d_{2,1} + c' = c \cdot b + d$. By induction hypothesis, we get $\sum r'_1[n] + \sum r'_2[n] = \sum r'[n] + c' - c_n \cdot b^{-n}$, for some $c_n \in \{-1, 0, 1, 2\}$, hence we have

$$\begin{aligned}
\sum r_1[n+1] + \sum r_2[n+1] &= b^{-1} d_{1,1} + b^{-1} \sum r'_1[n] + b^{-1} d_{2,1} + b^{-1} \sum r'_2[n] \\
&= b^{-1}(d + bc - c') + b^{-1} \left(\sum r'[n] + c' - b^{-n} c_n \right) \\
&= b^{-1} d + \sum r'[n] + c - b^{-(n+1)} c_n \\
&= \sum r[n] + c - b^{-(n+1)} c_n
\end{aligned}$$

From the proof of soundness we observe that the fact that the carry is forced by corules to belong to $\{-1, 0, 1, 2\}$ is essential: it assures that the sequence $(b^{-n} c_n)_{n \in \mathbb{N}}$ converges to 0. Indeed, if we let c range over $\mathbb{Z}$, then the inference system becomes unsound; for instance, it would be possible to build the following infinite proof for $\mathsf{add}(\overline{0}, \overline{0}, \overline{0}, 1)$ where all nodes clearly have a finite proof in $\mathcal{I}^{\mathsf{add}} \cup \mathcal{I}^{\mathsf{add}}_{\mathsf{co}}$, and, hence, $\langle \mathcal{I}^{\mathsf{add}}, \mathcal{I}^{\mathsf{add}}_{\mathsf{co}} \rangle \vdash_\nu \mathsf{add}(\overline{0}, \overline{0}, \overline{0}, 1)$ would hold, but $[\![\overline{0}]\!] + [\![\overline{0}]\!] \neq [\![\overline{0}]\!] + 1$:

$$\dfrac{\dfrac{\dfrac{\vdots}{\mathsf{add}(\overline{0}, \overline{0}, \overline{0}, b^2)}}{\mathsf{add}(\overline{0}, \overline{0}, \overline{0}, b^1)}}{\mathsf{add}(\overline{0}, \overline{0}, \overline{0}, b^0)}$$

3.5.2 Distances and shortest paths on weighted graphs

In Section 3.1, we have shown a first example concerning graphs, defining the judgement $v \to^\star N$, stating that N is the set of nodes reachable from v in the graph. Essentially, the proposed definition performs a visit of the graph following all possible, even infinite, paths. The same pattern can be adopted to solve more complex problems. For instance, in this section we will deal with distances between nodes in a *weighted graph*.

Let us introduce the notion of weights for graphs. Recall that (cf. p.18) we modelled a graph G by its adjaciency function $G : V \to \wp(V)$, where V is a finite set of nodes. With this representation, the set of edges is the set $E \subseteq V \times V$ defined by $E = \{\langle v, u \rangle \in V \times V \mid u \in G(v)\}$. We will often write vu for an edge $\langle v, u \rangle \in E$. A weight function is a function $w : E \to \mathbb{N}$. Here we consider natural numbers as codomain, however we could have considered any other set of non-negative numbers. Hence, a weighted graph is a graph G together with a weight function w.

A path from v_0 to v_n in G is a sequence of nodes $\alpha = v_0 \dots v_n$ with $n \geq 0$, such that, for all $i \in 1..n$, $v_{i-1} v_i \in E$. The empty path starting from the node v

$$\dfrac{\dfrac{\dfrac{}{\mathsf{dist}_G(e,e,0)} \quad \dfrac{\vdots}{\dfrac{}{\mathsf{dist}_G(a,e,\delta_1)}}}{\mathsf{dist}_G(b,e,\delta_1)}}{\mathsf{dist}_G(a,e,\delta_1)}\ \delta_1 \leq 5}{\mathsf{dist}_G(c,e,1+\delta_1)}$$

$$\dfrac{\dfrac{}{\mathsf{dist}_G(e,d,\infty)}\ G(e)=\emptyset \quad \dfrac{\vdots}{\dfrac{}{\mathsf{dist}_G(a,e,\delta_2)}}}{\mathsf{dist}_G(b,d,\delta_2)}}{\mathsf{dist}_G(a,d,\delta_2)}$$

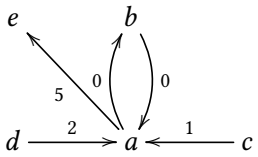
FIGURE 3.4 Infinite proof trees for $\mathsf{dist}_G(c,e,1+\delta_1)$ and $\mathsf{dist}_G(a,d,\delta_2)$

to itself is the sequence v of length 1. In a weighted graph G, the weight of a path α, denoted by $w(\alpha)$, is the sum of the weights of the edges determined by α, that is, $w(v) = 0$ and $w(vu\beta) = w(v,u) + w(u\beta)$. Note that in general the weight of a path α is different from its length, denoted by $\|\alpha\|$, defined as the number of edges (counting repetitions) determined by the path, that is, $\|v\| = 0$ and $\|vu\beta\| = 1 + \|u\beta\|$. The distance between nodes v and u is defined as the minimum weight of a path connecting v to u, it is infinite if no such path exists. Below we show the inference system with corules defining the judgement $\mathsf{dist}_G(v,u,\delta)$ on a weighted graph, where $\delta \in \mathbb{N} \cup \{\infty\}$.

$$(\text{D-E})\ \frac{}{\mathsf{dist}_G(v,v,0)} \qquad (\text{CO-D})\ \frac{}{\mathsf{dist}_G(v,u,\infty)}\ v \neq u$$

$$(\text{D-S})\ \frac{\mathsf{dist}_G(v_1,u,\delta_1) \quad \dots \quad \mathsf{dist}_G(v_k,u,\delta_k)}{\mathsf{dist}_G(v,u,\delta)} \quad \begin{array}{l} v \neq u \\ G(v) = \{v_1,\dots,v_k\} \\ \delta = \inf\{w(vv_i) + \delta_i \mid i \in 1..k\} \end{array}$$

In order to show that we cannot simply consider the coinductive interpretation of the above inference system, and therefore we need corules, let us consider the following weighted graph:



If we interpreted the inference system coinductively, we could derive judgements like $\mathsf{dist}_G(c,e,\delta)$ for any $\delta \in 1..6$ or $\mathsf{dist}_G(a,d,\delta)$ for any $\delta \in \mathbb{N} \cup \{\infty\}$, as shown in Figure 3.4.

The issue here is the cycle that, having total weight equal to 0, allows us to build cyclic proofs without increasing the value of δ. Therefore, the coaxiom is needed to filter out such proofs. Indeed, it is easy to see that it is not possible to build a finite proof tree for the judgements proved in Figure 3.4 starting from the coaxiom.

Now we will sketch a proof of correctness. Let us denote by $\langle \mathcal{I}^{\mathsf{dist}}, \mathcal{I}^{\mathsf{dist}}_{\mathsf{co}} \rangle$ the inference system with corules defined above. Assume a weighted graph G and, for all nodes v, u, in G, denote by $\delta(v, u)$ the distance from v to u, that is, $\delta(v, u) = \inf\{w(\alpha) \mid \alpha \text{ is a path from } v \text{ to } u\}$. We can formulate the correctness statement as follows: $\langle \mathcal{I}^{\mathsf{dist}}, \mathcal{I}^{\mathsf{dist}}_{\mathsf{co}} \rangle \vdash_\nu \mathsf{dist}_G(v, u, \delta)$ holds iff $\delta = \delta(v, u)$. In the following we say a judgement $\mathsf{dist}_G(v, u, \delta)$ is correct, if $\delta = \delta(v, u)$.

COMPLETENESS The completeness statement is as follows:

$\langle \mathcal{I}^{\mathsf{dist}}, \mathcal{I}^{\mathsf{dist}}_{\mathsf{co}} \rangle \vdash_\nu \mathsf{dist}_G(v, u, \delta(v, u))$ holds, for all nodes v, u.

The proof is by bounded coinduction (cf. Proposition 3.27). Let us consider a judgement $\mathsf{dist}_G(v, u, \delta)$ where $\delta = \delta(v, u)$. If $v = u$, then $\delta = 0$ and so the judgement is the consequence of (D-E), which has no premises. Suppose $v \neq u$. It is easy to check that $\delta(v, u) = \inf\{w(vv') + \delta(v', u) \mid v' \in G(v)\}$, hence, the judgement $\mathsf{dist}_G(v, u, \delta)$ is the consequence of (D-S), with premises $\mathsf{dist}_G(v', u, \delta(v', u))$, for all $v' \in G(v)$.

In order to show the boundedness condition, we have to build a finite proof tree for $\mathsf{dist}_G(v, u, \delta(v, u))$ in $\mathcal{I}^{\mathsf{dist}} \cup \mathcal{I}^{\mathsf{dist}}_{\mathsf{co}}$. We generalise this statement as follows: if there is no path from v to u, then $\mathcal{I}^{\mathsf{dist}} \cup \mathcal{I}^{\mathsf{dist}}_{\mathsf{co}} \vdash_\mu \mathsf{dist}_G(v, u, \infty)$, and, for all paths α from v to u, $\mathcal{I}^{\mathsf{dist}} \cup \mathcal{I}^{\mathsf{dist}}_{\mathsf{co}} \vdash_\mu \mathsf{dist}_G(v, u, w(\alpha))$. If there is no path from v to u, then $v \neq u$, hence we can derive $\mathsf{dist}_G(v, u, \infty)$ by (CO-D). In the other case, consider a path α from v to u. We proceed by induction on the length of α. If $\|\alpha\| = 0$, then $v = u$ and $w(\alpha) = 0$, hence we can apply (D-E). If $\|\alpha\| = n + 1$, then $\alpha = vv'\beta$ with $\|v'\beta\| = n$, $v' \in G(v)$ and $w(\alpha) = w(vv') + w(v'\beta)$. By induction hypothesis, we get that $\mathcal{I}^{\mathsf{dist}} \cup \mathcal{I}^{\mathsf{dist}}_{\mathsf{co}} \vdash_\mu \mathsf{dist}_G(v', u, w(v'\beta))$ holds, then we get $\mathcal{I}^{\mathsf{dist}} \cup \mathcal{I}^{\mathsf{dist}}_{\mathsf{co}} \vdash_\mu \mathsf{dist}_G(v, u, \delta)$ by applying (D-S) to $\mathsf{dist}_G(v', u, w(v'\beta))$ and $\mathsf{dist}_G(v'', u, \infty)$, for each $v'' \in G(v) \setminus \{v'\}$, which are derivable by (CO-D).

SOUNDNESS The soundness statement is as follows:

if $\langle \mathcal{I}^{\mathsf{dist}}, \mathcal{I}^{\mathsf{dist}}_{\mathsf{co}} \rangle \vdash_\nu \mathsf{dist}_G(v, u, \delta)$, then $\delta = \delta(v, u)$.

For all $n \in \mathbb{N}$, we denote by $\delta_n(v, u)$ the minimum weight of a path α of length less than or equal to n from v to u, that is, $\delta_n(v, u) = \inf\{w(\alpha) \mid \|\alpha\| \leq n,\ \alpha \text{ is a path from } v \text{ to } u\}$. For all $n \in \mathbb{N}$, we have $\delta_{n+1}(v, u) = \inf\{w(vv') + \delta_n(v', u) \mid v' \in G(v)\}$.

It is easy to see that, if $\delta(v, u) \leq \delta \leq \delta_n(v, u)$, for all $n \in \mathbb{N}$, then $\delta = \delta(v, u)$. Therefore, to check the thesis, we just have to prove that, for all $n \in \mathbb{N}$, if $\mathsf{dist}_G(v, u, \delta)$ has an approximated proof tree of level n, then $\delta(v, u) \leq \delta \leq \delta_n(v, u)$. It is easy to prove by induction on rules in $\mathcal{I}^{\mathsf{dist}} \cup \mathcal{I}^{\mathsf{dist}}_{\mathsf{co}}$ that, if $\mathcal{I}^{\mathsf{dist}} \cup \mathcal{I}^{\mathsf{dist}}_{\mathsf{co}} \vdash_\mu \mathsf{dist}_G(v, u, \delta)$, then $\delta(v, u) \leq \delta$. Since, if $\mathsf{dist}_G(v, u, \delta)$ has an approximated proof tree of level n, then $\mathcal{I}^{\mathsf{dist}} \cup \mathcal{I}^{\mathsf{dist}}_{\mathsf{co}} \vdash_\mu \mathsf{dist}_G(v, u, \delta)$ holds, we have only to prove the second inequality, namely, $\delta \leq \delta_n(v, u)$. The proof is by induction on n.

Case: 0 If $v = u$, then we have applied (D-E), hence $\delta = 0 = \delta_0(v, u)$. If $v \neq u$, then $\delta_0(v, u) = \inf \emptyset = \infty$, hence $\delta \leq \delta_0(v, u)$.

$$\text{(P-E)}\ \frac{}{\mathsf{minPath}_G(v, v, v, 0)} \qquad \text{(P-N)}\ \frac{}{\mathsf{minPath}_G(v, u, \bot, \infty)}\ \begin{array}{l} v \neq u \\ G(v) = \emptyset \end{array}$$

$$\text{(CO-P)}\ \frac{}{\mathsf{minPath}_G(v, u, \bot, \infty)}\ v \neq u$$

$$\text{(P-S)}\ \frac{\mathsf{minPath}_G(v_1, u, \alpha_1, \delta_1) \quad \ldots \quad \mathsf{minPath}_G(v_k, u, \alpha_k, \delta_k)}{\mathsf{minPath}_G(v, u, v\alpha_i, w(vv_i) + \delta_i)}\ \begin{array}{l} v \neq u \\ G(v) = \{v_1, \ldots, v_k\} \neq \emptyset \\ i = \arg\min\{w(vv_h) + \delta_h \mid h \in 1..k\} \end{array}$$

FIGURE 3.5 Inference system with corules for $\mathsf{minPath}_G(v, u, \alpha, \delta)$.

Case: $n + 1$ If $v = u$, then we have applied (D-E), hence $\delta = 0 = \delta_{n+1}(v, u)$. If $v \neq u$, then we have applied (D-S), hence, for all $v' \in G(v)$, $\mathsf{dist}_G(v', u, \delta_{v'})$ has an approximated proof tree of level n and $\delta = \inf\{w(vv') + \delta_{v'} \mid v' \in G(v)\}$. By induction hypothesis, we have, for all $v' \in G(v)$, $\delta_{v'} \leq \delta_n(v', u)$, hence, since $\delta_{n+1}(v, u) = \inf\{w(vv') + \delta_n(v', u) \mid v' \in G(v)\}$, we get the thesis.

The notion of distance is tightly related to paths in a graph G. Actually, from the above proofs, it is easy to see that a proof tree for a judgement $\mathsf{dist}_G(v, u, \delta)$ explores all possible paths from v to u in the graph in order to compute the distance. Therefore, in some sense, it also finds the shortest path from v to u. Hence, with a slight variation of the inference system for the distance, we can get an inference system for the judgement $\mathsf{minPath}_G(v, u, \alpha, \delta)$ stating that α is the shortest path from v to u with weight δ. We add to paths a special value $\bot$ that represents the absence of paths between two nodes, with the assumption that $v\bot = \bot$. The definition is reported in Figure 3.5, where $\arg\min\{x_1, \ldots, x_n\}$ denotes the index $i \in 1..n$ of the minimum among $x_1, \ldots, x_n$.

3.5.3 Mixing induction and coinduction

Even if an inference system can define together different types of judgements, in the standard framework they are all interpreted in the same way, that is, either all inductively or all coinductively. As a consequence, we cannot define together judgements requiring different interpretations. Typically, this issue is addressed by stratification: judgements and rules are divided in different strata, each of them containing only judgements and rules requiring the same interpretation, and the definition in each stratum can only depend on upper strata, that is, no cyclic dependency is allowed. This is for instance the strategy adopted by Simon et al. (2007) to define the semantis of co-logic programming. Stratification of course is not always possible and, in fact, it is the same as providing different definitions of each stratum.

We have already seen (cf. Corollary 3.15) that an inductive inference system corresponds to a generalised inference system with no corules, while a coinductive one corresponds to the case where there is a coaxiom for each judgement in the universe; however, between these two extremes, corules offer many other possibilities, thus allowing for a finer control on the semantics of the defined judgements. Since there exist cases where different judgements need to be defined together, but require different interpretations, see, *e.g.*, examples by Simon et al. (2006, 2007), Ancona (2013), and Basold and Komendantskaya (2016), corules may be employed to provide the correct definition in terms of a single inference system with no stratification. However, the interaction between corules and standard rules is not trivial at all, and, as we will see, unexpected behaviours can happen; hence special care is required to get from the inference system the intended meaning of judgements.

In order to see this, let us consider the judgement $\mathsf{path}_0(t)$, where t is an infinite (ordered) tree[1] over $\{0, 1\}$, represented as an infinite term of shape $\mathsf{tree}(n, l)$, where $n \in \{0, 1\}$ is the root of the tree, and l is the infinite list of its direct subtrees. Then, $\mathsf{path}_0(t)$ holds iff there exists a path in t, starting from the root, containing just 0s. For instance, if t_1 and t_2 are the trees defined by the syntactic equations

$$t_1 = \mathsf{tree}(0, l_1) \qquad l_1 = t_2{:}t_1{:}l_1 \qquad t_2 = \mathsf{tree}(0, l_2) \qquad l_2 = \mathsf{tree}(1, l_1){:}l_2$$

then we expect $\mathsf{path}_0(t_1)$ to hold, but not $\mathsf{path}_0(t_2)$.

To define path_0, we introduce an auxiliary judgement $\mathsf{isin}_0(l)$ testing whether an infinite list l of trees contains a tree t such that $\mathsf{path}_0(t)$ holds. Intuitively, we expect path_0 and isin_0 to be interpreted coinductively and inductively, respectively; this reflects the fact that path_0 checks a property universally quantified over an infinite sequence (a *safety* property in the terminology of concurrent systems): all the elements of the path must be equal to 0; on the contrary, isin_0 checks a property existentially quantified over an infinite sequence (a *liveness* property in the terminology of concurrent systems): the list must contain a tree t with a specific property (that is, $\mathsf{path}_0(t)$ must hold). Driven by this intuition, one could be tempted to define the following inference system with corules for all judgements of shape $\mathsf{path}_0(t)$, and no corules for judgements of shape $\mathsf{isin}_0(l)$:

$$\frac{\mathsf{isin}_0(l)}{\mathsf{path}_0(\mathsf{tree}(0, l))} \qquad \frac{}{\mathsf{path}_0(t)} \qquad \frac{\mathsf{path}_0(t)}{\mathsf{isin}_0(t{:}l)} \qquad \frac{\mathsf{isin}_0(l)}{\mathsf{isin}_0(t{:}l)}$$

Unfortunately, because of the mutual recursion between isin_0 and path_0, the inference system above does not capture the intended behaviour: $\mathsf{isin}_0(l)$ is derivable for every infinite list of trees l, even when l does not contain a tree t with an infinite path starting from its root containing just 0s. Indeed, the coaxiom we added is not really restrictive, because it allows the predicate path_0 to be coinductive, but, since isin_0 directly depends on path_0, it is allowed to be coinductive as well.

1 For the purpose of this example, we only consider trees with infinite depth and branching.

To overcome this problem, we can break the mutual dependency between judgements, replacing the judgement isin_0 with the more general one isin, such that $\mathsf{isin}(t, l)$ holds iff the infinite list l contains the tree t. Consequently, we can define the following inference system with corules:

$$(\textsc{ptho})\ \frac{\mathsf{isin}(t, l) \quad \mathsf{path}_0(t)}{\mathsf{path}_0(\mathsf{tree}(0, l))} \qquad (\textsc{co-ptho})\ \frac{}{\mathsf{path}_0(t)} \qquad (\textsc{in-h})\ \frac{}{\mathsf{isin}(t, t{:}l)}$$

$$(\textsc{in-t})\ \frac{\mathsf{isin}(t, l)}{\mathsf{isin}(t, t'{:}l)}$$

Now the semantics of the system corresponds to the intended one, since now isin does not depend on path_0, hence the corules do not influence the semantics of isin, which remains inductive as expected. Nevertheless, the semantics is well-defined without the need of stratifying the definitions into two separate inference systems.

Using proof trees and the proof techniques provided in Section 3.4, we can sketch a proof of correctness. Let $\langle \mathcal{I}^{\mathsf{path}_0}, \mathcal{I}_{\mathsf{co}}^{\mathsf{path}_0} \rangle$ be the inference system with corules defined above and $\mathcal{S}^{\mathsf{path}_0}$ be the set of judgements defined as follows: $\mathsf{path}_0(t) \in \mathcal{S}^{\mathsf{path}_0}$ iff t represents a tree with an infinite path of just 0s starting from its root, and $\mathsf{isin}(t, l) \in \mathcal{S}^{\mathsf{path}_0}$ iff l contains t. Then, the correctness statement is as follows: $\langle \mathcal{I}^{\mathsf{path}_0}, \mathcal{I}_{\mathsf{co}}^{\mathsf{path}_0} \rangle \vdash_\nu j$ iff $j \in \mathcal{S}^{\mathsf{path}_0}$, with $j = \mathsf{path}_0(t)$ or $j = \mathsf{isin}(t, l)$.

COMPLETENESS The proof is by bounded coinduction (Proposition 3.27). We first show that the set $\mathcal{S}^{\mathsf{path}_0}$ is a post-fixed point, that is, it is consistent w.r.t. the inference rules. Indeed, if t has an infinite path of 0s, then it has necessarily shape $\mathsf{tree}(0, l)$, where l must contain a tree t' with an infinite path of 0s. Hence, $\mathsf{path}_0(t)$ is the consequence of (PTHO) with premises $\mathsf{isin}(t', l) \in \mathcal{S}^{\mathsf{path}_0}$ and $\mathsf{path}_0(t') \in \mathcal{S}^{\mathsf{path}_0}$. If an infinite list contains a tree t, then it has necessarily shape $t'{:}l$ where, either $t = t'$, and hence $\mathsf{isin}(t, t{:}l)$ is the consequence of (IN-H), with no premises, or t belongs to l, and hence $\mathsf{isin}(t, t'{:}l)$ is the consequence of (IN-T) with premise $\mathsf{isin}(t, l) \in \mathcal{S}^{\mathsf{path}_0}$.

We now show boundedness, that is, if $j \in \mathcal{S}^{\mathsf{path}_0}$, then j has a finite proof tree in $\mathcal{I}^{\mathsf{path}_0} \cup \mathcal{I}_{\mathsf{co}}^{\mathsf{path}_0}$. For the elements of shape $\mathsf{path}_0(t)$ it suffices to directly apply (CO-PTHO). For the elements of shape $\mathsf{isin}(t, l)$ where t belongs to l, the thesis follows by a straightforward induction on the position of t in l.

SOUNDNESS We first observe that the only finite proof trees in $\mathcal{I}^{\mathsf{path}_0} \cup \mathcal{I}_{\mathsf{co}}^{\mathsf{path}_0}$ that can be constructed for $\mathsf{isin}(t, l)$ use only standard rules (IN-H) and (IN-T), hence $\langle \mathcal{I}^{\mathsf{path}_0}, \mathcal{I}_{\mathsf{co}}^{\mathsf{path}_0} \rangle \vdash_\nu \mathsf{isin}(t, l)$ holds iff there exists a finite proof tree for $\mathsf{isin}(t, l)$ in $\mathcal{I}^{\mathsf{path}_0}$. Then, soundness for judgements of shape $\mathsf{isin}(t, l)$ follows by a straightforward induction on rules in $\mathcal{I}^{\mathsf{path}_0}$.

For the elements of shape $\mathsf{path}_0(t)$ we observe that any proof tree for $\mathsf{path}_0(t)$ in $\mathcal{I}^{\mathsf{path}_0}$ must be infinite, because there are no axioms for path_0 and in (PTHO) path_0 is referred in the premises. Then, it is easy to check that, if $\mathsf{path}_0(t)$ has an infinite proof tree, then it contains an infinite path containing just 0s. This is

because such infinite derivation is built applying infinitely many times (PTHO), which, each time, checks that the root is 0 and then access a direct subtree.

4

Discussion

Inference systems are a general and versatile framework to define possibly recursive judgements, which is well-known and widely used. However, standard semantics of inference systems is, in a sense, rigid: either inductive (the least one) or coinductive (the greatest one), but what can we do if we need something in the middle? One may wonder whether this is a real issue, but the examples we have provided in Chapter 3 and throughout this thesis show that there are many interesting cases where we need an interpretation that is neither the least nor the greatest one. This is the reason why we introduce an extension of inference systems, enabling more flexible interpretations, to support formal reasoning even in cases where standard (co)inductive semantics is not enough.

The core of the proposed more general framework are corules, which are special rules that, specified together with traditional rules, allow us to control their interpretation. From a model-theoretic perspective, corules are used to restrict the universe on which we take the greatest fixed point of the inference operator associated with the inference system; while, from the proof-theoretic perspective, corules impose additional conditions on (infinite) proof trees, filtering out some of them.

In this part we formally describe inference systems and their generalisaiton by corules, providing the formal tools used in the rest of the thesis. More in detail, Chapter 2 describes standard inference systems and their semantics both in model-theoretic and proof-theoretic terms. We provide all details on this well-known notions as we have not found in literature a sufficiently complete and rigorous treatement to develop our results. In particular, we frame standard equivalemce results between model-theoretic and proof-theoretic approaches in an abstract setting, relating trees and judgments by means of an adjunction. This setting is then used throughout the thesis to express and prove all other results of this kind.

In Chapter 3, we describe the generalisation of inference systems by corules. At a first glance, corules are used to restrict the set of rules that have to be interpreted coinductively, that is, the interpretation of an inference system with corules $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$ is defined as the coinductive interpretation of the inference system obtained by restricting $\mathcal{I}$ to rules with conclusion in $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$. To prove this construction provides indeed an interpretation of $\mathcal{I}$, namely, a fixed point of the associated inference operator, we first construct the needed fixed point, the bounded fixed point, in the standard lattice-theoretic setting,

which is a combination of least and greatest fixed point constructions, and then prove that the interpretation of $\langle \mathcal{I}, \mathcal{I}_{co} \rangle$ is an instance of such a fixed point. From a proof-theoretic perspective, we describe several proof-theoretic characterisation of the interpretation of an inference system with corules, based on a combination of well-founded and non-well-founded proof trees and on approximated proof trees. Finally, relying on these model-theoretic and proof-theoretic characterisation, we describe proof-techniques to reason with corules and apply them on several examples.

4.1 Related work

As already mentioned, inference systems (Aczel, 1977; Leroy and Grall, 2009; Sangiorgi, 2011) are a widely used framework to express possibly recursive definitions of predicated by means of rules, providing rigorous, but fairly simple, semantic foundations. Although inference systems have been introduced to deal with inductive definitions, in the last two decades several authors have focused on their coinductive interpretation.

Cousot and Cousot (1992) define divergence of programs by coinductive interpretation of an inference system that extends the big-step operational semantics. The same approach is followed by other authors, such as Hughes and Moran (1995) and Leroy and Grall (2009). Leroy and Grall (2009) analyse two kinds of coinductive big-step operational semantics for the call-by-value λ-calculus, and study their relationships with the small-step and denotational semantics, and their suitability for compiler correctness proofs. Coinductive big-step semantics is used as well to reason about cyclic objects stored in memory by Milner and Tofte (1991) and Leroy and Rouaix (1998), and to prove type soundness in Java-like languages by Ancona (2012, 2014). Coinductive inference systems are also considered in the context of type analysis and subtyping for object-oriented languages by Ancona and Lagorio (2009) and Ancona and Corradi (2014).

On the programming language side, coinduction is adopted to provide primitives helping the programmer dealing with infinite objects. Examples can be found both in logic programming, such as works by Simon et al. (2006, 2007) and Johann, Komendantskaya, and Komendantskiy (2015), and in functional programming by Hagino (1987) and Bird and Wadler (1988). Recently, other approaches have been proposed to support coinduction in programming languages in a more flexible way. We can find contributions in all most popular paradigms: logic paradigm, by Ancona, 2013; Mantadelis, Rocha, and Moura, 2014, functional paradigm, by Jeannin, Kozen, and Silva (2013, 2017) and object-oriented paradigm, by Ancona and Zucca (2012, 2013). As a consequence, these proposals are more focused on operational aspects, and their corresponding implementation issues. Our work originates from some of these operational models, which are closely related to each other, namely, works by Ancona and Zucca (2012, 2013) and Ancona (2013). Indeed, as just said, these models introduce some flexibility when defining predicates and functions recursively

on non-well-founded structures, and our first aim has been to provide a more abstract view of these approaches.

One of the distinguishing features of the theory of inference systems is that it does not consider any specific syntax, allowing a purely semantic analysis of rule-based definitions. On the other hand, in the literature, there are several proposals of formal systems supporting induction and coinduction which provide a syntactic approach to recursive definitions in general and rule-based definitions in particular. Momigliano and Tiu (2003), Brotherston (2005), Gacek, Miller, and Nadathur (2008), and Brotherston and Simpson (2011) propose logical calculi with possibly recursive definitions of predicates represented as equations that can be interpreted either inductively or coinductively. Doumane (2017) studies infinitary proof systems for logics supporting recursive predicates by fixed point combinators, but only in a propositional setting. Basold (2018) describes simple and dependent type theories with mixed inductive-coinductive types again via fixed point combinators, thus supporting recursive definitions of predicates. Finally, another approach to support coinduction in proof systems or type theories is by the later modality (Bizjak et al., 2016; Basold, 2018; Basold, Komendantskaya, and Li, 2019), which ensures soundness by guarding recursive references. Any of the above formal systems can, in principle, be used to provide the syntactic counterpart of inference systems. Furthermore, when both induction and coinduction are supported, also flexible coinduction can, in principle, be encoded, since it is defined as a combination of induction and coinduction.

4.2 Future work

There are several directions for further developments. A first compelling topic is enhancing proof techniques for corules, trying to extend proof techniques known for coinduction to this generalised framework. More specifically, the notion of bounded coinduction is a combination of a standard coinductive proof method (establishing a suitable post-fixed point) and a separate inclusion in a domain determined by rules and corules. The coinductive part is amenable to up-to techniques (Pous, 2007; Pous and Sangiorgi, 2012; Pous, 2016), which may help to simplify proofs using bounded coinduction, and parametric coinduction (Hur et al., 2013), which may be useful in formalisation in a proof assistant.

Another important goal is to provide the support for corules in a proof assistant, such as Agda (*The Agda Reference Manual*) or Coq (*The Coq Reference Manual*), to have a tool to mechanize and certify proofs. In dependent type theories supporting inductive and coinductive types (Hagino, 1987; Abel et al., 2013; Abel and Pientka, 2013; Basold, 2018), like the one at the basis of Agda, we can implement predicates defined by inference systems with corules just applying the definition: we can use a coinductive type, representing possibly infinite proof trees, which internally uses an inductive type to require each node to have finite proof tree with corules. We have analysed this possibility, using Agda, in a master thesis (Ciccone, 2019). What would be interesting is

to hide this construction, in such a way that the programmer has only to care about specifying rules and corules, leaving everything else to the engine. We are currently working on an Agda implementation of inference systems with corules, where we have types modelling sets of rules and corules and then a type construction taking an inference system with corules and producing a type modelling its interpretation. This requires the user to write rules and corules in a slightly unusual way, hence a futher development would be to allow the user writing definitions in a more familiar syntax and then compile it to produce Agda code.

Another question concerns the expressive power of this framework. Here by expressive power we mean "how many" subsets of the universe we can characterise. At the level considered in this thesis to develop the theory, this question is not significant: indeed, any subset of the universe can be expressed by an inference system consisting of one axiom for each judgement in such set, for which all interpretations are equivalent. However, this sounds not very useful, since, to define a subset, we use the subset itself. Actually, inference systems, and hence inference systems with corules, are never used in the form they are regarded in the development of the meta-theory, but, rather, they are expressed using a *finite set of meta-rules*, leaving implicit the step from meta-rules to plain rules, which, instead, are considered in the meta-theory. At this level, the above question becomes meaningful. Hence, to investigate expressive power in the appropriate setting, we should define what is an inference system in terms of meta-rules. To this end, interesting starting points could be the works by Momigliano and Tiu (2003) and Brotherston and Simpson (2011), which discuss proof systems for first-order logics with a notion of inductive and/or coinductive definitions.

Another source of inspiration to address the expressiveness issue could be computability theory, in particular the arithmetical hierarchy, which provides a tool to classify subsets of natural numbers depending on how much it is difficult to check membership. Restricting to a countable universe and under suitable conditions on inference rules, we conjecture that inductively definable sets are those in Σ^0_1 (recursively enumerable sets), coinductively definable sets are those in Π^0_1, and sets definable by an inference system with corules are those in Π^0_2, namely, they are more "difficult" than the other two classes. Ancona and Dovier (2015) have taken this perspective to characterise the coinductive semantics of definite logic programs, which are a particular syntactic instance of inference systems, showing that it is not recursively enumerable.

Another interesting direction is to investigate variants/improvements of the model, to avoid unexpected behaviours like those discussed in Sections 3.5.3 and 7.2. The first possible improvement is to move from sets and subsets to families and subfamilies. In this way it would be possible to design a model able to force a corule to be applicable only to some judgements. This is particularly useful when an inference system defines multiple judgements, for instance the following one:

$$\frac{p(x)}{p(x)} \qquad \frac{q(x)}{p(x)} \qquad \frac{q(x)}{q(x+1)} \qquad \frac{}{q(x)}$$

where $x \in \mathbb{N} + \{\infty\}$. Here, as there is no corule for judgements of shape $p(x)$, we expect it to be provable only by finite derivations, hence the first rule should be irrelevant and so $p(x)$ should hold iff $q(x)$ holds. Therefore, since $q(x)$ holds iff $x = \infty$, because the only derivation is the following infinite one

$$\dfrac{\dfrac{\dfrac{\vdots}{q(\infty)}}{q(\infty)}}{q(\infty)}$$

we expect that only $p(\infty)$ holds. However, in the current model, as all judgements are treated the same way, this is not true, since we have the following proof trees, which are correct for any $x \in \mathbb{N} + \{\infty\}$:

$$\dfrac{\dfrac{\dfrac{\vdots}{p(x)}}{p(x)}}{p(x)} \qquad\qquad \dfrac{\dfrac{\overline{}}{q(x)}}{p(x)}$$

In other words, rules for $p(x)$ are interpreted coinductively, even if there is no explicit corule for $p(x)$.

Another possible variant of the model is to make the connection between rule and corules stronger. Roughly, at the moment the semantics is constructed in two completely independent steps, hence corules can be freely applied even to judgements that at the end are not in the constructed fixed point. This could raise issues in some cases (cf. the example in Section 7.2) and we conjecture it could be solved by slightly changing the fixed point construction.

PART II

Infinite behaviour by big-step semantics

5

Big-step semantics: an operational perspective

The semantics of programming languages or software systems specifies, for each program/system configuration, its final result, if any. In the case of non-existence of a final result, there are two possibilities:

- either the computation stops with no final result, and there is no means to compute further: *stuck computation*,
- or the computation never stops: *non-termination*.

There are two main styles to define operationally a semantic relation: the *small-step* style (Plotkin, 1981, 2004), on top of a transition relation representing single computation steps, or directly by a set of rules as in the *big-step* style (Kahn, 1987). Within a small-step semantics it is straightforward to make the distinction between stuck and non-terminating computations, while a typical drawback of the big-step style is that they are not distinguished (no judgement is derived in both cases). Actually, in big-step style, it is not even clear what a computation is, because the only available notion we have is derivability of judgements, which does not convey the dynamics of computation.

For this reason, even though big-step semantics is generally more abstract, and sometimes more intuitive to design and therefore to debug and extend, in the literature much more effort has been devoted to study the meta-theory of small-step semantics, providing properties, and related proof techniques. Notably, the *soundness* of a type system (typing prevents stuck computation) can be proved by *progress* and *subject reduction*, also called *type preservation*, (Wright and Felleisen, 1994). Note that soundness cannot even be *expressed* with respect to a big-step semantics, since non-termination and stuckness are confused, as they are both modelled by the absence of a final result.

Our quest in this chapter is to develop a meta-theory of big-step operational semantics, to enable formal reasoning also on non-terminating computations. More precisely we will address the following problems:

1. Defining, in a formal way, *computations* in a given arbitrary big-step semantics.
2. According to this definition, providing constructions *yielding an extended version of a given arbitrary big-step semantics*, where the difference between stuckness and non-termination is made explicit.

3. Providing a general proof technique by identifying *three sufficient conditions* on the original big-step rules to prove soundness of a predicate.

All these three points rely on the same fundamental cornerstone: a general definition of big-step semantics. Such a definition captures the essential features of a big-step semantics, independently from the particular language or system.

To address Item 1, we rely on the intuition that every big-step semantics implicitly defines an evaluation algorithm and we identify computations in the big-step semantics with computations of such algorithm. Formally, we extend the big-step semantics to model *partial evaluations*, representing intermediate states of the evaluation process, and we formalise the evaluation algorithm by a transition relation between such intermediate states. In this way, we can easily distinguish stuck and non-terminating computations, showing that this distinction is actually present, but hidden, in any big-step semantics. In a sense, this definition makes explicit the operational nature of big-step semantics, which is not so evident as for small-step one.

Constructions in Item 2 provide extended big-step semantics able to distinguish between stuck and non-terminating computations, as obtained by Item 1, but abstracting away single computation steps. More in detail, starting from an arbitrary big-step judgment $c \Rightarrow r$ that evaluates *configurations* c into *results* r, the first construction produces an enriched judgement $c \Rightarrow_{\mathsf{tr}} r_{\mathsf{tr}}$ where r_{tr} is either a pair $\langle t, r\rangle$ consisting of a finite trace t and a result r, or an infinite trace σ. Finite and infinite traces model the (finite or infinite) sequences of all the configurations encountered during the evaluation. In this way, by interpreting coinductively the rules of the extended semantics, an infinite trace models divergence (whereas no result corresponds to stuck computation). Furthermore, we will show that, by using coaxioms, we can get rid of traces, modelling divergence just by a judgmeent $c \Rightarrow \infty$. The second construction is in a sense dual. It is the general version of the well-known technique presented in Exercise 3.5.16 by Pierce (2002) of adding a special result wrong explicitly modelling stuck computations (whereas no result corresponds to divergence). We will show that these constructions are correct, proving that they represent the intended class of computations as defined in Item 1.

Item 3's three sufficient conditions are *local preservation*, $\exists$*-progress*, and $\forall$*-progress*. For *proving* the result that the three conditions actually ensure soundness, we crucially rely on the extended big-step semantics of Item 2, since otherwise, as said above, we could not even express the property.

However, the three conditions deal only with the original rules of the given big-step semantics. This means that, practically, in order to use the technique there is no need to deal with the meta-theory (computations and extended semantics), exactly as happens for the progress and subject reduction technique for small-step semantics. This implies, in particular, that our approach does *not* increase the original number of rules. Moreover, the sufficient conditions are checked only on *single rules*, hence neither induction nor coinduction is needed. In a sense, they make explicit elementary fragments of the soundness proof,

which we carry out once and for all (cf. Theorems 5.38 and 5.41), embedding such semantic-dependent fragments in a semantic-independent (co)inductive proof. Even though this is not exploited in this thesis, this form of *locality* enables *modularity*, in the sense that adding a new rule implies adding the corresponding proof fragment only.

We support our approach by presenting several examples, demonstrating that: on the one hand, soundness proofs can be easily rephrased in terms of our technique, that is, by directly reasoning on big-step rules; on the other hand, our technique is essential when the property to be checked (for instance, well-typedness) is *not preserved* by intermediate computation steps, whereas it holds for the final result. On a side note, our examples concern type systems, but the meta-theory we present in this work holds for any predicate.

Actually, we can express two flavours of soundness, depending on whether we make explicit stuckness or non-termination. In the former case we express *soundness-must*, which is the notion of soundness we have considered so far, preventing all stuck computations, while in the latter case we express *soundness-may*, a weaker notion only ensuring *the existence of* a non-stuck computation. Of course, this distinction is relevant only in presence of non-determinism, otherwise the two notions coincide. We define a proof technique for soundness-may as well, showing it is correct. In the end, it should be noted that *we define soundness with respect to a big-step semantics within a big-step formulation*, without resorting to a small-step style (indeed, the extended semantics are themselves big-step).

The rest of the chapter is organised as follows. Section 5.1 provides a definition of big-step semantics. Section 5.2 introduces partial evaluation trees and a transition relation between them, modelling the evaluation algorithm guided by rules and defines computations in big-step semantics as possibly infinite sequences of steps in such transition relation. In this way we get a reference semantic model. Section 5.3 define two constructions extending a given big-step semantics: one, based on traces, which explicitly models diverging computations and another, which explicitly models stuck computations. Section 5.4 defines a third construction, modelling divergence just as a special result, by using appropriate corules. Section 5.5 shows how we can express two flavours of soundness against big-step semantics and provide proof techniques to show this property. Finally, Section 5.6 shows how to use the proof technique on several examples.

5.1 Defining big-step semantics

As mentioned in the introduction, the corner stone of this chapter is a formalisation of "what a big-step semantics'is, that captures its essential features, subsuming a large class of examples. This enables a general formal reasoning on an arbitrary big-step semantics.

DEFINITION 5.1 : A *big-step semantics* is a triple $\langle C, R, \mathcal{R} \rangle$ where:

- C is a set of *configurations* c.
- R is a set of *results* r. We define *judgments* $j \equiv c \Rightarrow r$, meaning that configuration c evaluates to result r. Set $C(j) = c$ and $R(j) = r$.
- $\mathcal{R}$ is a set of *(big-step) rules* ρ of shape

$$\frac{j_1 \quad \dots \quad j_n}{c \Rightarrow r} \qquad \text{also written in } \textit{inline format}\text{: } \mathsf{rule}(j_1 \dots j_n,\ c,\ r)$$

  where $j_1 \dots j_n$, with $n \geq 0$, is a sequence of premises. Set $C(\rho) = c$, $R(\rho) = r$ and, for $i \in 1..n$, $C(\rho, i) = C(j_i)$ and $R(\rho, i) = R(j_i)$.

We will use the inline format, more concise and manageable, for the development of the meta-theory, e.g., in constructions.

Big-step rules, as defined above, are very much like inference rules as in Definition 2.1, but they carry slightly more structure with respect to them. Notably, premises are a sequence rather than a set, that is, they are ordered and there can be repeated premises. Such additional structure, however, does not affect derivability, namely, the inference operator and so the interpretations of such rules. Therefore, given a big-step semantics $\langle C, R, \mathcal{R} \rangle$, slightly abusing the notation, we denote by $\mathcal{R}$ the inference system obtained by forgetting such additional structure, and define, as usual, the *semantic relation* as the inductive interpretation of $\mathcal{R}$. As usual (cf. Definition 2.6), we will write $\mathcal{R} \vdash_\mu c \Rightarrow r$ when the judgment $c \Rightarrow r$ is derivable in $\mathcal{R}$.

Even though the additional structure of big-step rules does not affect the semantic relation they define, it is crucial to develop the meta-theory, allowing abstract reasoning about an arbitrary big-step semantics. It will be used in all results in this chapter: to define computations in big-step semantics, then to provide constructions yelding extended semantics able to distinguish stuck and diverging computations and, finally, to define proof techniques for soundness. Indeed, as premises are a sequence, we know in which order configurations in the premises should be evaluated.

In practice, the (infinite) set of rules $\mathcal{R}$ is described by a finite set of meta-rules, each one with a finite number of premises. As a consequence, the number of premises of rules is not only finite but *bounded*. Since we have no notion of meta-rule, we model this feature (relevant in the following) as an explicit assumption:

ASSUMPTION 5.1 (Bounded premises (BP)): For a big-step semantics $\langle C, R, \mathcal{R} \rangle$, there exists $b \in \mathbb{N}$ such that, for each $\rho = \mathsf{rule}(j_1 \dots j_n,\ c,\ r)$, $n \leq b$.

We end this section by illustrating the above definitions and conditions on a simple example: a λ-calculus with constants for natural numbers, successor and non-deterministic choice, shown in Figure 5.1. It is immediate to see this example as an instance of Definition 5.1:

- Configurations and results are expressions, and values, respectively.[1]

1 In general, configurations may include additional components and results are not necessarily particular configurations, see, *e.g.*, Section 5.6.2.

$$
\begin{array}{lcll}
e & ::= & x \mid v \mid e_1\, e_2 \mid \mathsf{succ}\, e \mid e_1 \oplus e_2 & \text{expression} \\
v & ::= & n \mid \lambda x.e & \text{value}
\end{array}
$$

$$
(\textsc{val})\ \frac{}{v \Rightarrow v} \qquad (\textsc{app})\ \frac{e_1 \Rightarrow \lambda x.e \quad e_2 \Rightarrow v_2 \quad e[v_2/x] \Rightarrow v}{e_1\, e_2 \Rightarrow v}
$$

$$
(\textsc{succ})\ \frac{e \Rightarrow n}{\mathsf{succ}\, e \Rightarrow n+1} \qquad (\textsc{choice})\ \frac{e_i \Rightarrow v}{e_1 \oplus e_2 \Rightarrow v}\ \ i = 1,2
$$

(VAL) $\mathsf{rule}(\varepsilon,\ v,\ v)$
(APP) $\mathsf{rule}(e_1 \Rightarrow \lambda x.e\ e_2 \Rightarrow v_2\ e[v_2/x] \Rightarrow v,\ e_1\, e_2,\ v)$
(SUCC) $\mathsf{rule}(e \Rightarrow n,\ \mathsf{succ}\, e,\ n+1)$
(CHOICE) $\mathsf{rule}(e_i \Rightarrow v,\ e_1 \oplus e_2,\ v)\ i = 1,2$

FIGURE 5.1 Example of big-step semantics

- To have the set of (meta-)rules in our required shape, abbreviated in inline format in the bottom section of the figure, we have only to assume an order on premises of rule (APP).

REMARK: The order of premises chosen for rule (APP) in Figure 5.1 formalises the evaluation strategy for an application $e_1\ e_2$ where first (1) evaluates e_1, then (2) checks that the value of e_1 is a λ-abstraction, finally (3) evaluates e_2. That is, left-to-right evaluation with early error detection. Other strategies can be obtained by choosing a different order or by adjusting big-step rules. Notably, right-to-left evaluation (3)-(1)-(2) can be expressed by just swapping the first two premises, that is:

$$(\textsc{app-r})\ \mathsf{rule}(e_2 \Rightarrow v_2\ e_1 \Rightarrow \lambda x.e\ e[v_2/x] \Rightarrow v,\ e_1\, e_2,\ v)$$

Left-to-right evaluation with late error detection (1)-(3)-(2) can be expressed as follows:

$$(\textsc{app-late})\ \mathsf{rule}(e_1 \Rightarrow v_1\ e_2 \Rightarrow v_2\ v_1 \Rightarrow \lambda x.e\ e[v_2/x] \Rightarrow v,\ e_1\, e_2,\ v)$$

We can even opt for a non-deterministic approach by taking more than one rule among (APP), (APP-R) and (APP-LATE). As said above, these different choices do not affect the semantic relation inductively defined by the inference system, which is always the same. However, they will affect computations and thus the extended semantics distinguishing stuck computation and non-termination. Indeed, if the evaluation of e_1 and e_2 is stuck and non-terminating, respectively, we should obtain a stuck computation with rule (APP) and non-termination with rule (APP-R); further, if e_1 evaluates to a natural constant and e_2 diverges, we should obtain a stuck computation with rule (APP) and non-termination with rule (APP-LATE).

In summary, to see a typical big-step semantics as an instance of our definition, it is enough to identify configurations and results and to assume an order (or more than one) on premises.

5.2 Computations in big-step semantics

Intuitively, the evaluation of a configuration c is a *dynamic* process and, as such, it may either successfully terminate producing the final result, or get stuck, or never terminate. However, a big-step semantics just tells us whether a configuration c evaluates to a certain result r, without describing the dynamics of such evaluation process. This is nice, because it allows us to abstract away details about intermediate states in the evaluation process, but it makes quite difficult to reason about concepts like non-termination and stuckness, since they refer to computations and we do not even know what a computation is in a big-step semantics.

In this section, we show that, given a big-step semantics as defined in Definition 5.1, we can recover the dynamics of the evaluation, by defining *computations*, which, in a sense, are implicit in a big-step specification. To this end, we extend the big-step semantics, so that we can represent partial (or incomplete) evaluations, modelling intermediate states of the evaluation process. Then, we model the dynamics by a transition relation beween such partial evaluations, hence, as usual, a computation will be a (possibly infinite) sequence of transitions.

Let us assume a big-step semantics $\langle C, R, \mathcal{R}\rangle$. As said above, the first step is to extend such semantics to model partial evaluations. To this end, first of all, we introduce a special result ?, so that a judgment $c \Rightarrow ?$ (called *incomplete*, whereas a judgment $c \Rightarrow r$ is *complete*) means that the evaluation of c is not completed yet. Set $R_? = R + \{?\}$ whose elements are ranged over by $r_?$. We now define an augmented set of rules $\mathcal{R}_?$ to properly handle the new result ?:

DEFINITION 5.2 (Rules for partial evaluation): The set of rules $\mathcal{R}_?$ is obtained from $\mathcal{R}$ by adding the following rules:

START RULES For each configuration $c \in C$, define rule $\mathsf{ax}_?(c)$ as $\dfrac{}{c \Rightarrow ?}$.

PARTIAL RULES For each rule $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r)$ in $\mathcal{R}$, index $i \in 1..n$, and $r_? \in R_?$, define rule $\mathsf{pev}_?(\rho, i, r_?)$ as

$$\frac{j_1 \quad \ldots \quad j_{i-1} \quad C(j_i) \Rightarrow r_?}{c \Rightarrow ?}$$

Intuitively, start rules allow us to begin the evaluation of any configuration, while partial rules allow us to partially apply a rule from $\mathcal{R}$ to derive a partial judgement. Note that the last premise of a partial rule can be either complete ($r_? \in R$) or incomplete ($r_? = ?$), in the latter case we also call it a *?-propagation* rule, since it propagates ? from premises to the conclusion.

It is important to observe that the construction described above yields a triple $\langle C, R_?, \mathcal{R}_?\rangle$, which is a big-step semantics according to Definition 5.1. In Figure 5.2 we report rules added by the construction in Definition 5.2 to the big-step semantics of the λ-calculus in Figure 5.1.

Given a big-step semantics $\langle C, R, \mathcal{R}\rangle$, using rules in $\mathcal{R}$, we can build trees called *evaluation trees*. Such trees are very much like proof trees for an inference

$$\frac{}{e \Rightarrow ?} \qquad \frac{e \Rightarrow v_?}{\mathsf{succ}\, e \Rightarrow ?} \qquad \frac{e_i \Rightarrow v_?}{e_1 \oplus e_2 \Rightarrow ?}\ i = 1, 2$$

$$\frac{e_1 \Rightarrow v_?}{e_1\ e_2 \Rightarrow ?} \qquad \frac{e_1 \Rightarrow \lambda x.e \quad e_2 \Rightarrow v_?}{e_1\ e_2 \Rightarrow ?} \qquad \frac{e_1 \Rightarrow \lambda x.e \quad e_2 \Rightarrow v_2 \quad e[v_2/x] \Rightarrow v_?}{e_1\ e_2 \Rightarrow ?}$$

FIGURE 5.2 Rules for ? for the λ-calculus in Figure 5.1.

system, as defined in Section 2.1 (cf. Definition 2.5), with the only difference that evaluation trees are *ordered* trees, because premises of big-step rules are a sequence. Roughly, an evaluation tree is an ordered tree with nodes labelled by semantic judgements, such that for each node labelled by $c \Rightarrow r$ with sequence of children $j_1, \ldots, j_n$, there is a rule $\mathsf{rule}(j_1 \ldots j_n,\ c,\ r)$ in $\mathcal{R}$.

An evaluation tree for $\langle C, R_?, \mathcal{R}_? \rangle$ is called a *partial evaluation tree*, as it can contain incomplete judgements. We say that a partial evaluation tree is *complete* if it only contains complete judgments, it is *incomplete* otherwise. Finite partial evaluation trees indeed model possibly incomplete evaluation of configurations, namely, the intermediate states of the evaluation process, because big-step rules can be partially applied. Hence, they are the fundamental building block, which will allow us to define computations in big-step semantics.

In the next subsection we will give a formal definition of (partial) evaluation trees, similar to the one used for proof trees (cf. Section 2.1.1). However, this formal definition is only needed to state some results and to carry out proofs in a rigorous way, and not to follow the rest of the chapter, hence the reader not interested in formal details can skip it, relying on the above semiformal definition.

5.2.1 The structure of partial evaluation trees

We give a formal account of (partial) evaluation trees, which is useful to state and prove technical results in the next sections. This development is based on the definition and properties of trees provided by Courcelle (1983), adjusted to our specific setting.

Set $\mathbb{N}_{>0}$ the set of positive natural numbers and $\mathcal{L}$ a set of labels. An *ordered tree* labelled in $\mathcal{L}$ is a partial function $\tau : \mathbb{N}_{>0}^{\star} \to \mathcal{L}$ such that $\mathsf{dom}(\tau)$ is not empty, and, for each $\alpha \in \mathbb{N}_{>0}^{\star}$ and $n \in \mathbb{N}_{>0}$, if $\alpha n \in \mathsf{dom}(\tau)$ then $\alpha \in \mathsf{dom}(\tau)$ and, for all $k \leq n$, $\alpha k \in \mathsf{dom}(\tau)$. Given an ordered tree τ and $\alpha \in \mathsf{dom}(\tau)$, set $\mathsf{br}_\tau(\alpha) = \sup\{n \in \mathbb{N} \mid \alpha n \in \mathsf{dom}(\tau)\}$ the *branching* of τ at α, and $\tau_{|\alpha}$ the *subtree* of τ rooted at α, that is, $\tau_{|\alpha}(\beta) = \tau(\alpha\beta)$, for all $\beta \in \mathbb{N}_{>0}^{\star}$. The *root* of τ is $\mathsf{r}(\tau) = \tau(\varepsilon)$ and obviously we have $\tau = \tau_{|\varepsilon}$. Finally, we write $\dfrac{\tau_1 \quad \ldots \quad \tau_n}{x}$ for the tree τ defined by $\tau(\varepsilon) = x$, and $\tau(i\alpha) = \tau_i(\alpha)$ for all $i \in 1..n$.

This definition of tree is very much like the one introduced in Section 2.1.1, and so notations are almost the same. There are, however, three main differences: these trees are ordered, each node has only finitely many children, and there can be two sibling nodes with the same label. These additional

features are essential here to properly deal with big-step semantics. Since in the following we will only deal with ordered trees, we will refer to them just as trees.

Let us assume a big-step semantics $\langle C, R, \mathcal{R}\rangle$. Assume also that labels in $\mathcal{L}$ are semantic judgments $c \Rightarrow r$, then we can define evaluation trees as follows:

DEFINITION 5.3 : A tree $\tau : \mathbb{N}^{\star}_{>0} \to \mathcal{L}$ is an *evaluation tree* in $\langle C, R, \mathcal{R}\rangle$, if, for each $\alpha \in \mathsf{dom}(\tau)$ with $\tau(\alpha) = c \Rightarrow r$, there is $\mathsf{rule}(\tau(\alpha 1) \dots \tau(\alpha \mathsf{br}_{\tau}(\alpha)),\ c,\ r) \in \mathcal{R}$.

Note that, starting from an evaluation tree τ, we can construct a proof tree, as defined in Definition 2.5, for the inference system denoted by $\mathcal{R}$, by forgetting the order on sibling nodes and removing duplicated children. Therefore, if τ is a finite evaluation tree with $\mathsf{r}(\tau) = c \Rightarrow r$, then $\mathcal{R} \vdash_{\mu} c \Rightarrow r$ holds.

DEFINITION 5.4 : A *partial evaluation tree* in $\langle C, R, \mathcal{R}\rangle$ is an evaluation tree in $\langle C, R_?, \mathcal{R}_?\rangle$.

The following proposition assures two key properties of partial evaluation trees. First, if there is some ?, then it is propagated to ancestor nodes. Second, for each level of the tree there is at most one ?. We set $|\alpha|$ the length of $\alpha \in \mathbb{N}^{\star}_{>0}$.

PROPOSITION 5.5 : Let τ be a partial evaluation tree, then the following hold:

1. for all $\alpha n \in \mathsf{dom}(\tau)$, if $R_?(\tau(\alpha n)) = ?$ then $R_?(\tau(\alpha)) = ?$.
2. for all $n \in \mathbb{N}$, there is at most one $\alpha \in \mathsf{dom}(\tau)$ with $|\alpha| = n$ such that $R_?(\tau(\alpha)) = ?$.

Proof: To prove Item 1, we just have to note that the only rules having a premise j with $R_?(j) = ?$ are ?-propagation rules, which also have conclusion j' with $R_?(j') = ?$; hence the thesis is immediate. To prove Item 2, we proceed by induction on n. For $n = 0$, there is only one $\alpha \in \mathbb{N}^{\star}_{>0}$ with $|\alpha| = 0$ (the empty sequence), hence the thesis is trivial. Consider $\alpha = \alpha' k \in \mathsf{dom}(\tau)$ with $|\alpha| = n + 1$. If $R_?(\tau(\alpha)) = ?$, then, by Item 1, $R_?(\tau(\alpha')) = ?$, and, by induction hypothesis, α' is the only sequence of length n in $\mathsf{dom}(\tau)$ with this property. Therefore, another node $\beta \in \mathsf{dom}(\tau)$, with $|\beta| = n + 1$ and $R_?(\tau(\beta)) = ?$, must satisfy $\beta = \alpha' h$ for some $h \in \mathbb{N}_{>0}$; hence, since τ is a partial evaluation tree, $\tau(\alpha)$ and $\tau(\beta)$ are two premises of the same rule with ? as result, thus they must coincide, since all rules in $\mathcal{R}_?$ have at most one premise with ?. □

COROLLARY 5.6 : Let τ be a partial evaluation tree, then $R_?(\mathsf{r}(\tau)) \in R$ if and only if τ is complete.

We can define a relation[2], denoted by $\sqsubseteq$, on trees labelled by possibly incomplete judgements, as follows:

2 This is a slight variation of similar relations on trees considered by Courcelle (1983) and Dagnino (2019).

DEFINITION 5.7 : Let τ and τ' be trees labelled by possibly incomplete semantic judgements. Define $\tau \sqsubseteq \tau'$ if and only if $\mathrm{dom}(\tau) \subseteq \mathrm{dom}(\tau')$ and, for all $\alpha \in \mathrm{dom}(\tau)$, $C(\tau(\alpha)) = C(\tau'(\alpha))$ and $R_?(\tau(\alpha)) \in R$ implies $\tau_{|\alpha} = \tau'_{|\alpha}$.

Intuitively, $\tau \sqsubseteq \tau'$ means that τ' can be obtained from τ by adding new branches or replacing some ?s with results. We use $\sqsubset$ for the strict version of $\sqsubseteq$. Note that, if $\tau \sqsubseteq \tau'$, then, for all $\alpha \in \mathbb{N}^\star_{>0}$, $\tau'(\alpha)$ is *more defined* than $\tau(\alpha)$, because, either $\tau(\alpha)$ is undefined, or $\tau(\alpha)$ is incomplete and $C(\tau(\alpha)) = C(\tau'(\alpha))$, or $\tau(\alpha) = \tau'(\alpha)$.

It is easy to check that $\sqsubseteq$ is a partial order and, if $\tau \sqsubseteq \tau'$, then, for all $\alpha \in \mathrm{dom}(\tau)$, $\tau_{|\alpha} \sqsubseteq \tau'_{|\alpha}$. The following proposition shows some, less trivial, properties of $\sqsubseteq$.

PROPOSITION 5.8 : The following properties hold:

1. for all trees τ and τ', if $\tau \sqsubseteq \tau'$ and $R_?(\mathrm{r}(\tau)) \in R$, then $\tau = \tau'$
2. for each increasing sequence $(\tau_i)_{i\in\mathbb{N}}$ of trees, there is a least upper bound $\tau = \bigsqcup \tau_n$.

Proof: Item 1 is immediate by definition of $\sqsubseteq$. To prove Item 2, first note that, since for all $n \in \mathbb{N}$, $\tau_n \sqsubseteq \tau_{n+1}$, for all $\alpha \in \mathbb{N}^\star_{>0}$ we have that, for all $n \in \mathbb{N}$, if $\tau_n(\alpha)$ is defined, then, for all $k \geq n$, $C(\tau_k(\alpha)) = C(\tau_n(\alpha))$, and, if $R_?(\tau_n(\alpha)) \in R$, then $\tau_k(\alpha) = \tau_n(\alpha)$. Hence, for all $n \in \mathbb{N}$, there are only three possibilities for $\tau_n(\alpha)$: it is either undefined, or equal to $c \Rightarrow ?$, or equal to $c \Rightarrow r$, where the configuration is the same. Let us denote by k_α the least index where $\tau_n(\alpha)$ is most defined, hence, for all $n \geq k_\alpha$, we have that $\tau_n(\alpha) = \tau_{k_\alpha}(\alpha)$. Then, consider a tree τ defined by $\tau(\alpha) = \tau_{k_\alpha}(\alpha)$. It is easy to check that $\mathrm{dom}(\tau) = \bigcup_{n\in\mathbb{N}} \mathrm{dom}(\tau_n)$. We now check that, for all $n \in \mathbb{N}$, $\tau_n \sqsubseteq \tau$. For all $\alpha \in \mathrm{dom}(\tau_n)$, we have $\alpha \in \mathrm{dom}(\tau)$ and we distinguish two cases:

- if $\tau_n(\alpha) = c \Rightarrow ?$, then $k_\alpha \geq n$, hence, since $\tau_n \sqsubseteq \tau_{k_\alpha}$, we get $C(\tau(\alpha)) = C(\tau_{k_\alpha}(\alpha)) = C(\tau_n(\alpha)) = c$
- if $\tau_n(\alpha) = c \Rightarrow r$, then $k_\alpha \leq n$, hence, since $\tau_{k_\alpha} \sqsubseteq \tau_n$, we get $C(\tau(\alpha)) = C(\tau_{k_\alpha}(\alpha)) = C(\tau_n(\alpha)) = c$, thus we have only to check that $\tau_{n\,|\alpha} = \tau_{|\alpha}$. That is easy, because, for all $\beta \in \mathrm{dom}(\tau_{n\,|\alpha})$, we have $\tau_{n\,|\alpha}(\beta) = \tau_n(\alpha\beta) = c' \Rightarrow r'$, hence $k_{\alpha\beta} \geq n$, hence $\tau_{|\alpha}(\beta) = \tau(\alpha\beta) = \tau_{k_{\alpha\beta}}(\alpha\beta) = \tau_n(\alpha\beta)$, as needed.

This proves that τ is an upper bound of the sequence, we have still to prove that it is the least one. To this end, let τ' be an upper bound of the sequence: we have to show that $\tau \sqsubseteq \tau'$. Since τ' is an upper bound, for all $n \in \mathbb{N}$ we have $\mathrm{dom}(\tau_n) \subseteq \mathrm{dom}(\tau')$, hence $\mathrm{dom}(\tau) \subseteq \mathrm{dom}(\tau')$, and, especially, for all $\alpha \in \mathbb{N}^\star_{>0}$ we have $\tau_{k_\alpha} \sqsubseteq \tau'$. Hence, for all $\alpha \in \mathrm{dom}(\tau)$, we have $C(\tau(\alpha)) = C(\tau_{k_\alpha}(\alpha)) = C(\tau'(\alpha))$, and, if $R_?(\tau(\alpha)) = r$, since $\tau_{k_\alpha} \sqsubseteq \tau$ and $\tau_{k_\alpha} \sqsubseteq \tau'$, we have $\tau_{k_\alpha\,|\alpha} = \tau_{|\alpha}$ and $\tau_{k_\alpha\,|\alpha} = \tau'_{|\alpha}$, hence $\tau_{|\alpha} = \tau'_{|\alpha}$, as needed. □

Obviously, this relation restricts to partial evaluation trees and, more importantly, the set of partial evaluation trees is closed with respect to least upper bound for $\sqsubseteq$, as the next proposition shows.

PROPOSITION 5.9 : For each increasing sequence $(\tau_n)_{n\in\mathbb{N}}$ of partial evaluation trees, the least upper bound $\bigsqcup \tau_n$ is a partial evaluation tree as well.

Proof: Set $\tau = \bigsqcup \tau_n$, and recall from Proposition 5.8 (2) that $\tau(\alpha) = \tau_{k_\alpha}(\alpha)$, where $k_\alpha \in \mathbb{N}$ is the least index n where $\tau_n(\alpha)$ is most defined. Note that, for all $\alpha \in \mathsf{dom}(\tau)$, $\mathsf{br}_\tau(\alpha)$ is finite, since, by definition of τ, we have $\mathsf{br}_\tau(\alpha) = \sup\{\mathsf{br}_{\tau_n}(\alpha) \mid n \in \mathbb{N}\}$, and this value is bounded because $\mathsf{br}_{\tau_n}(\alpha)$ is the number of premises of a rule, which is bounded by Assumption 5.1. Then, since $\mathsf{br}_\tau(\alpha)$ is finite, there is an index $n \in \mathbb{N}$ such that $\mathsf{br}_\tau(\alpha) = \mathsf{br}_{\tau_n}(\alpha)$ and, in particular, this holds for all $n \geq k_{\alpha\mathsf{br}_\tau(\alpha)}$. Set $n = \max\{k_\alpha, k_{\alpha 1}, \ldots, k_{\alpha\mathsf{br}_\tau(\alpha)}\}$, hence $n \geq k_{\alpha\mathsf{br}_\tau(\alpha)}$ and $\tau : n(\alpha) = \tau(\alpha)$ and $\tau_n(\alpha i) = \tau(\alpha i)$, for all $i \in 1..\mathsf{br}_\mathsf{r}(\alpha)$. Therefore, $\langle \tau(\alpha 1) \ldots \tau(\alpha\mathsf{br}_\tau(\alpha)), \tau(\alpha)\rangle = \langle \tau_n(\alpha 1) \ldots \tau_n(\alpha\mathsf{br}_\tau(\alpha)), \tau_n(\alpha)\rangle \in \mathcal{R}_?$, since τ_n is a partial evaluation tree. Thus, by Definition 5.4, τ is a partial evaluation tree. □

As already mentioned, finite partial evaluation trees model possibly incomplete evaluations. Then, the relation $\sqsubseteq$ models refinement of the evaluation, because if $\tau \sqsubseteq \tau'$, where τ and τ' are finite partial evaluation trees, τ' is "more detailed" than τ. In a sense, $\sqsubseteq$ on finite partial evaluation trees abstracts the process of evaluation itself, as we will make precise in the next section.

What about infinite trees? Similarly to what we have discussed in the introduction, there are many infinite partial evaluation trees which are difficult to interpret. For instance, using rules in Figure 5.1 and Figure 5.2, we can construct the following infinite tree for all $v_?$, where $\Omega = (\lambda x.x\,x)\,(\lambda x.x\,x)$:

$$\dfrac{\dfrac{}{\lambda x.x\,x \Rightarrow \lambda x.x\,x} \quad \dfrac{}{\lambda x.x\,x \Rightarrow \lambda x.x\,x} \quad \dfrac{\vdots}{\Omega = (x\,x)[\lambda x.x\,x/x] \Rightarrow v_?}}{\Omega \Rightarrow v_?}$$

Among all such trees there are some "good" ones, we call them *well-formed*. Well-formed infinite partial evaluation trees arise as limits of strictly increasing sequences of finite partial evaluation trees, hence, in a sense, they model the limit of the evaluation process. namely, non-termination.

DEFINITION 5.10 : An infinite partial evaluation tree τ is *well-formed* if, for all $n \in \mathbb{N}$, there is $\alpha \in \mathsf{dom}(\tau)$ such that $|\alpha| = n$ and $R_?(\tau(\alpha)) = ?$, and, for all $\alpha \in \mathsf{dom}(\tau)$, if $R_?(\tau(\alpha)) \in R$, then $\tau_{|\alpha}$ is finite.

Informally, this means that a well-formed tree contains a unique infinite path, which is entirely labelled by incomplete judgments, hence all its complete subtrees are necessarily finite. Then, we can prove the following result:

PROPOSITION 5.11 : The following properties hold:

1. for each strictly increasing sequence $(\tau_n)_{n\in\mathbb{N}}$ of finite partial evaluation trees, the least upper bound $\bigsqcup \tau_n$ is infinite and well-formed;
2. for each well-formed infinite partial evaluation tree τ, there is a strictly increasing sequence $(\tau_n)_{n\in\mathbb{N}}$ of finite partial evaluation trees such that $\tau = \bigsqcup \tau_n$.

Proof: To prove point 1, set $\tau = \bigsqcup \tau_n$, then, by Proposition 5.9, we have that τ is a partial evaluation tree, hence we have only to check that it is infinite and well-formed. Since the sequence is strictly increasing, we have that, for all $n \in \mathbb{N}$, there is $h > n$ such that $\mathsf{dom}(\tau_n) \subset \mathsf{dom}(\tau_h)$, namely, there is $\alpha \in \mathsf{dom}(\tau_h)$ such that $\alpha \notin \mathsf{dom}(\tau_n)$. This can be proved by induction on the number of ? in τ, which is finite since τ_n is finite, noting that, if $\mathsf{dom}(\tau_n) = \mathsf{dom}(\tau_{n+1})$, since $\tau_n \sqsubset \tau_{n+1}$, there is at least one node $\alpha \in \mathsf{dom}(\tau_n)$ such that $R_?(\tau_n(\alpha)) = ?$ and $R_?(\tau_{n+1}(\alpha)) = r$. Therefore, $\mathsf{dom}(\tau) = \bigcup_{n\in\mathbb{N}} \mathsf{dom}(\tau_n)$ is infinite, that is, τ is infinite. To show that τ is well-formed, first recall that, for all $\alpha \in \mathsf{dom}(\tau)$, k_α is the least n such that $\tau_n(\alpha)$ is most defined. Note that, for all $\alpha \in \mathsf{dom}(\tau)$ such that $\tau(\alpha) = c \Rightarrow r$, since $\tau_{k_\alpha} \sqsubseteq \tau$ and $\tau_{k_\alpha}(\alpha) = \tau(\alpha)$, we get, by definition of $\sqsubseteq$, $\tau_{k_\alpha}{}_{|_\alpha} = \tau_{|_\alpha}$, hence $\tau_{|_\alpha}$ is finite. Then, we still have only to prove that, for each $n \in \mathbb{N}$, there is $\alpha \in \mathsf{dom}(\tau)$ such that $|\alpha| = n$ and $R_?(\tau(\alpha)) = ?$. We proceed by induction on n. For $n = 0$, we have $R_?(\mathsf{r}(\tau)) = ?$, since, otherwise, we would have $R_?(\mathsf{r}\tau_{k_\varepsilon}) = r$, hence, by Proposition 5.8 (1), we would get $\tau_{k_\varepsilon} = \tau_{k_\varepsilon+1}$ which is not possible, because the sequence is strictly increasing. Now, by induction hypothesis, we know there is $\alpha \in \mathsf{dom}(\tau)$ such that $|\alpha| = n$ and $R_?(\tau(\alpha)) = ?$. By Proposition 5.5, for all $\beta \in \mathsf{dom}(\tau)$ with $\beta = \alpha' h$ and $\alpha' \neq \alpha$, we have $R_?(\tau(\beta)) \in R$, because, if $R_?(\tau(\beta)) = ?$, then also $R_?(\tau(\alpha')) = ?$ and this implies $\alpha' = \alpha$, which is absurd, thus, for all such β, we have $\tau_{|_\beta}$ is finite, as we have just proved. Hence, we can focus on children of α, splitting cases over $\mathsf{br}_\tau(\alpha)$. If $\mathsf{br}_\tau(\alpha) = 0$, then α has no children and so τ is finite, which is absurd. If $h = \mathsf{br}_\tau(\alpha) > 0$, then, if $R_?(\tau(\alpha h)) \in R$, since τ is a partial evaluation tree, we get $R_?(\tau(\alpha h')) \in R$ for all $h' \leq h$, hence τ is finite, which is absurd, thus $R_?(\tau(\alpha h)) = ?$, as needed. Therefore, τ is well-formed as needed.

To prove point 2, for all $n \in \mathbb{N}$, consider the partial evaluation tree τ_n defined as follows: let $\alpha_n \in \mathsf{dom}(\tau)$ be the node such that $|\alpha_n| = n$ and $R_?(\tau(\alpha_n)) = ?$ (which exists as τ is well-formed and is unique thanks to Proposition 5.5 (2)), then define $\tau_n(\beta) = \tau(\beta)$ for all $\beta \neq \alpha_n\beta'$, with $\beta' \in \mathbb{N}^+_{>0}$, and undefined otherwise. We have $\tau_n \sqsubseteq \tau_{n+1}$, since, by Proposition 5.5 (1), $\alpha_{n+1} = \alpha_n i$ for some $i \in \mathbb{N}_{>0}$. Finally, by construction, we have $\tau = \bigsqcup \tau_n$, as needed. □

This important result will be used in the next sections to prove correctness of extended big-step semantics explicitly modelling divergence.

5.2.2 The transition relation

As already mentioned, finite partial evaluation trees nicely model intermediate states in the evaluation process of a configuration. We now make this precise by defining a transition relation $\longrightarrow_{\mathcal{R}}$ between them, such that, starting from the initial partial evaluation tree $\dfrac{}{c \Rightarrow ?}$, we derive a sequence where, intuitively, at each step we detail the evaluation. In this way, a sequence ending with a complete tree (a tree containing no ?) models succesfully terminating computation, whereas an infinite sequence (tending to an infinite partial evaluation tree) models divergence, and a sequence reaching an incomplete tree which cannot further move models a stuck computation.

The one-step transition relation $\longrightarrow_{\mathcal{R}}$ is inductively defined by the rules in Figure 5.3. Transitions are actually defined between *annotated* partial evaluation trees, that is, partial evaluation tree where each node is explicitly associated with the rule in $\mathcal{R}_?$ used to derive it from its children. This additional information is somehow redundant and only needed to make the definition clearer, hence in the following we will omit such annotations. In the figure, $\#\rho$ denotes the number of premises of ρ, and $\mathsf{r}\tau$ the root of τ. Finally, $\sim_i$ is the *equality up-to an index* of rules, defined below:

DEFINITION 5.12 : Let $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r)$ and $\rho' = \mathsf{rule}(j'_1 \ldots j'_m,\ c',\ r')$ be rules in $\mathcal{R}$. Then, for any index $i \in 1..\min(n, m)$, define $\rho \sim_i \rho'$ if and only if

- $c = c'$,
- for all $k < i$, $j_k = j'_k$, and
- $C(j_i) = C(j'_i)$.

Intuitively, this means that rules ρ and ρ' model the same computation until the i-th configuration included.

Intuitively, each transition step makes "less incomplete" the partial evaluation tree. Notably, transition rules apply only to nodes labelled by incomplete judgements ($c \Rightarrow ?$), whereas subtrees whose root is a complete judgement ($c \Rightarrow r$) cannot move. In detail:

- If the last applied rule is $\mathsf{ax}_?(c)$, we have to find a rule ρ with c in the conclusion and, if it has no premises we just return $R(\rho)$ as result, otherwise we start evaluating the first premise of such rule.
- If the last applied rule is $\mathsf{pev}_?(\rho, i, r)$, then all subtrees are complete, hence, to continue the evaluation, we have to find another rule ρ', having, for each $k \in 1..i$, as k-th premise the root of τ_k. Then there are two possibilities: if there is an $i+1$-th premise, we start evaluating it, otherwise, we return $R(\rho')$ as result.
- If the last applied rule is a propagation rule $\mathsf{pev}_?(\rho, i, ?)$, then we simply propagate the step made by τ_i (the last subtree).

$$(\text{TR-1}) \quad {}_{(\text{ax}_?(c))}\frac{}{c \Rightarrow ?} \longrightarrow_{\mathcal{R}} {}_{(\rho)}\frac{}{c \Rightarrow r} \quad \begin{array}{l} \#\rho = 0 \\ C(\rho) = c \\ R(\rho) = r \end{array}$$

$$(\text{TR-2}) \quad {}_{(\text{ax}_?(c))}\frac{}{c \Rightarrow ?} \longrightarrow_{\mathcal{R}} {}_{(\text{pev}_?(\rho,1,?))}\frac{c' \Rightarrow ?}{c \Rightarrow ?} \quad \begin{array}{l} \#\rho > 0 \\ C(\rho) = c \\ C(\rho, 1) = c' \end{array}$$

$$(\text{TR-3}) \quad {}_{(\text{pev}_?(\rho,i,r))}\frac{\tau_1 \ \ldots \ \tau_i}{c \Rightarrow ?} \longrightarrow_{\mathcal{R}} {}_{(\rho')}\frac{\tau_1 \ \ldots \ \tau_i}{c \Rightarrow r'} \quad \begin{array}{l} \rho' \sim_i \rho \\ R(\rho', i) = r \\ \#\rho' = i \\ R(\rho') = r' \end{array}$$

$$(\text{TR-4}) \quad {}_{(\text{pev}_?(\rho,i,r))}\frac{\tau_1 \ \ldots \ \tau_i}{c \Rightarrow ?} \longrightarrow_{\mathcal{R}} {}_{(\text{pev}_?(\rho',i+1,?))}\frac{\tau_1 \ \ldots \ \tau_i \ c' \Rightarrow ?}{c \Rightarrow ?} \quad \begin{array}{l} \rho' \sim_i \rho \\ R(\rho', i) = r \\ C(\rho', i+1) = c' \end{array}$$

$$(\text{TR-5}) \quad {}_{(\text{pev}_?(\rho,i,?))}\frac{\tau_1 \ \ldots \ \tau_{i-1} \ \tau_i}{c \Rightarrow ?} \longrightarrow_{\mathcal{R}} {}_{(\text{pev}_?(\rho,i,r_?))}\frac{\tau_1 \ \ldots \ \tau_{i-1} \ \tau_i'}{c \Rightarrow ?} \quad \tau_i \longrightarrow_{\mathcal{R}} \tau_i'$$

FIGURE 5.3 Transition relation between partial evaluation trees.

$$\frac{}{(\lambda x.x)\, n \Rightarrow ?} \longrightarrow_{\mathcal{R}} \frac{\lambda x.x \Rightarrow ?}{(\lambda x.x)\, n \Rightarrow ?} \longrightarrow_{\mathcal{R}} \frac{\lambda x.x \Rightarrow \lambda x.x}{(\lambda x.x)\, n \Rightarrow ?}$$

$$\longrightarrow_{\mathcal{R}} \frac{\lambda x.x \Rightarrow \lambda x.x \quad n \Rightarrow ?}{(\lambda x.x)\, n \Rightarrow ?} \longrightarrow_{\mathcal{R}} \frac{\lambda x.x \Rightarrow \lambda x.x \quad n \Rightarrow n}{(\lambda x.x)\, n \Rightarrow ?}$$

$$\longrightarrow_{\mathcal{R}} \frac{\lambda x.x \Rightarrow \lambda x.x \quad n \Rightarrow n \quad n \Rightarrow ?}{(\lambda x.x)\, n \Rightarrow ?} \longrightarrow_{\mathcal{R}} \frac{\lambda x.x \Rightarrow \lambda x.x \quad n \Rightarrow n \quad n \Rightarrow n}{(\lambda x.x)\, n \Rightarrow ?}$$

$$\longrightarrow_{\mathcal{R}} \frac{\lambda x.x \Rightarrow \lambda x.x \quad n \Rightarrow n \quad n \Rightarrow n}{(\lambda x.x)\, n \Rightarrow n}$$

FIGURE 5.4 The evaluation of $(\lambda x.x)\, n$ using $\longrightarrow_{\mathcal{R}}$ for rules in Figure 5.1.

In Figure 5.4 we report an example of evaluation of a term according to rules in Figure 5.1, using partial evaluation trees and $\longrightarrow_{\mathcal{R}}$.

As mentioned above, the definition of $\longrightarrow_{\mathcal{R}}$ given in Figure 5.3 nicely models as a transition system an interpreter driven by the big-step rules. In other words, the one-step transition relation between finite partial evaluation trees specifies an algorithm of incremental evaluation.[3] On the other hand, also the partial order relation $\sqsubseteq$ (cf. Definition 5.7) models a refinement relation between finite partial evauation trees, even if in a more abstract way. The next proposition formally proves that these two descriptions agree, namely, $\sqsubseteq$ is indeed an abstraction of $\longrightarrow_{\mathcal{R}}$.

PROPOSITION 5.13 : Let τ and τ' be finite partial evaluation trees, then the following hold:

1. if $\tau \longrightarrow_{\mathcal{R}} \tau'$ then $\tau \sqsubset \tau'$, and
2. if $\tau \sqsubseteq \tau'$ then $\tau \longrightarrow_{\mathcal{R}}^{\star} \tau'$.

3 Non-determinism can only be caused by intrinsic non-determinism of the big-step semantics, if any.

Proof: Point 1 can be easily proved by induction on the definition of $\longrightarrow_{\mathcal{R}}$. The proof of point 2 is by induction on τ', denote by IH the induction hypothesis. This is possible as τ' is finite by hypothesis. We can assume $R_?(\mathrm{r}(\tau)) = ?$, since in the other case, by Proposition 5.8 (1), we have $\tau = \tau'$, hence the thesis is trivial. We can further assume $R_?(\mathrm{r}\tau') = ?$, since, if $\tau' = \dfrac{\tau'_1 \quad \dots \quad \tau'_n}{c \Rightarrow r}$, then we always have $\tau'' = \dfrac{\tau'_1 \quad \dots \quad \tau'_n}{c \Rightarrow ?} \longrightarrow_{\mathcal{R}} \tau'$ and $\tau \sqsubseteq \tau''$. Now, if $\tau' = \dfrac{}{c \Rightarrow ?}$ (base case), then, since $\mathrm{dom}(\tau) \subseteq \mathrm{dom}(\tau')$ and $C(\mathrm{r}(\tau)) = C(\mathrm{r}\tau')$ by definition of $\sqsubseteq$, we have $\tau = \tau'$, hence the thesis is trivial.

Let us assume $\tau = \dfrac{\tau_1 \quad \dots \tau_k}{c \Rightarrow ?}$ and $\tau' = \dfrac{\tau'_1 \quad \dots \quad \tau'_i}{c' \Rightarrow ?}$, with, necessarily, $k \leq i$ and $c = c'$ by definition of $\sqsubseteq$. We have $\tau_h \sqsubseteq \tau'_n$, for all $h \leq k$, and by Proposition 5.5 (2), at most τ_k is incomplete, that is, for all $h < k$, τ_h is complete, namely, $R_?(\mathrm{r}\tau_h) \in R$, thus, by definition of $\sqsubseteq$, we have $\tau_h = \tau'_h$. Furthermore, since $\tau_k \sqsubseteq \tau'_k$, by IH, we get $\tau_k \longrightarrow^{\star}_{\mathcal{R}} \tau'_k$, hence $\tau \longrightarrow^{\star}_{\mathcal{R}} \tau'' = \dfrac{\tau'_1 \quad \dots \quad \tau'_k}{c \Rightarrow ?} \sqsubseteq \tau'$. We now show, concluding the proof, by arithmetic induction on $i - k$, that $\tau'' \longrightarrow^{\star}_{\mathcal{R}} \tau'$. If $i - k = 0$, hence $i = k$, we have $\tau'' = \tau'$, hence the thesis is immediate. If $i - k > 0$, hence $i > k$, setting $c'' = C(\mathrm{r}(\tau'_{k+1}))$, by IH, we get $\dfrac{}{c'' \Rightarrow ?} \longrightarrow^{\star}_{\mathcal{R}} \tau'_{k+1}$; moreover, again by Proposition 5.5 (2), we have $R_?(\mathrm{r}\tau'_k) \in R$, hence we get

$$\tau'' \longrightarrow_{\mathcal{R}} \frac{\tau'_1 \quad \dots \quad \tau'_k \quad c'' \Rightarrow ?}{c \Rightarrow ?} \longrightarrow^{\star}_{\mathcal{R}} \frac{\tau'_1 \quad \dots \quad \tau'_k \quad \tau'_{k+1}}{c \Rightarrow ?} = \hat{\tau}$$

Finally, by arithmetic induction hypothesis, we get $\hat{\tau} \longrightarrow^{\star}_{\mathcal{R}} \tau'$, as needed. □

We conclude this section by showing that the transition relation $\longrightarrow_{\mathcal{R}}$ agrees with the semantic relation (inductively) defined by $\mathcal{R}$, namely, the semantic relation captures exactly successful terminating computations in $\longrightarrow_{\mathcal{R}}$.

THEOREM 5.14 : $\mathcal{R} \vdash_{\mu} c \Rightarrow r$ iff $\dfrac{}{c \Rightarrow ?} \longrightarrow^{\star}_{\mathcal{R}} \tau$, where $\mathrm{r}(\tau) = c \Rightarrow r$.

Proof: $\mathcal{R} \vdash_{\mu} c \Rightarrow r$ implies $\dfrac{}{c \Rightarrow ?} \longrightarrow^{\star}_{\mathcal{R}} \tau$ where $\mathrm{r}(\tau) = c \Rightarrow r$. By definition, if $\mathcal{R} \vdash_{\mu} c \Rightarrow r$ holds, then there is a finite evaluation tree τ in $\mathcal{R}$ such that $\mathrm{r}(\tau) = c \Rightarrow r$. Since $\mathcal{R} \subseteq \mathcal{R}_?$ by Definition 5.2, τ is a (complete) partial evaluation tree as well; furthermore, $\dfrac{}{c \Rightarrow ?} \sqsubseteq \tau$, hence, by Proposition 5.13 (2), we get the thesis.

$\dfrac{}{c \Rightarrow ?} \longrightarrow^{\star}_{\mathcal{R}} \tau$ where $\mathrm{r}(\tau) = c \Rightarrow r$ implies $\mathcal{R} \vdash_{\mu} c \Rightarrow r$. Since $\mathrm{r}(\tau) = c \Rightarrow r$, by Corollary 5.6, τ is complete, hence, it is an evaluation tree in $\mathcal{R}$, thus $\mathcal{R} \vdash_{\mu} c \Rightarrow r$ holds. □

5.3 Extended big-step semantics: two constructions

In Section 5.2, we have just shown that, given a big-step semantics as in Definition 5.1, it is possible to define computations in such semantics, by deriving a transition relation which formally models the evaluation algorithm guided by the rules. In this way, we are able to distinguish stuck and non-terminating computations as in standard small-step semantics. This, in a sense, shows that such a distinction is *implicit* in a big-step semantics.

In this section, we aim at showing that we can make such distinction explicit directly by a big-step semantics, without introducing any transition relation modelling single computation steps. To this end, we describe two constructions that, starting from a big-step semantics, yield extended ones where non-terminating and stuck computations are explicitly distinguished. These two constructions are in some sense dual to each other, because one explicitly models non-termination, while the other one explicitly models stuckness, and they are based on well-know ideas: divergence is modelled by *traces*, as suggested by Leroy and Grall (2009), while stuckness by an additional special result, as proposed by Pierce (2002). The novel contribution is that, thanks to the general definition of big-step semantics in Section 5.1 (cf. Definition 5.1), we can provide *general* constructions working on an arbitrary big-step semantics, rather than discussing specific examples, as it is customary in the literature.

In the following, we assume a big-step semantics $\langle C, R, \mathcal{R} \rangle$.

5.3.1 Traces

The set of *traces* in the big-step semantics is the set C^∞ of finite and infinite sequences of configurations. Finite traces are ranged over by t, while infinite traces by σ.

The judgement of trace semantics has shape $c \Rightarrow_{\mathsf{tr}} r_{\mathsf{tr}}$, where $r_{\mathsf{tr}} \in Tr_R^C = (C^\star \times R) + C^\omega$, that is, r_{tr} is either a pair $\langle t, r_{\mathsf{tr}} \rangle$ of a finite trace and a result, modelling a converging computation, or an infinite trace σ, modelling divergence. Intuitively, traces t keep track of all the configurations visited during the evaluation, starting from c itself. To define the trace semantics, we construct, starting from $\mathcal{R}$, a new set of rules $\mathcal{R}_{\mathsf{tr}}$ as follows:

DEFINITION 5.15 (Rules for traces): The set of rules $\mathcal{R}_{\mathsf{tr}}$ consists of the following rules:

FINITE TRACE RULES For each $\rho = \mathsf{rule}(j_1 \dots j_n,\ c,\ r)$ in $\mathcal{R}$ and finite traces $t_1, \dots, t_n \in C^\star$, define rule $\mathsf{trace}(\rho,\ t_1, \dots, t_n)$ as

$$\frac{C(j_1) \Rightarrow_{\mathsf{tr}} \langle t_1, R(j_1) \rangle \quad \dots \quad C(j_n) \Rightarrow_{\mathsf{tr}} \langle t_n, R(j_n) \rangle}{c \Rightarrow_{\mathsf{tr}} \langle c t_1 \cdots t_n, r \rangle}$$

INFINITE TRACE RULES For each $\rho = \mathsf{rule}(j_1 \dots j_n,\ c,\ r)$ in $\mathcal{R}$, index $i \in 1..n$, finite traces $t_1, \dots, t_{i-1} \in C^\star$, and infinite trace $\sigma \in C^\omega$, define rule $\mathsf{trace}_\infty(\rho,\ i,\ t_1, \dots, t_{i-1},\ \sigma)$ as follows:

$$\frac{C(j_1) \Rightarrow_{\mathsf{tr}} \langle t_1, R(j_1) \rangle \quad \dots \quad C(j_{i-1}) \Rightarrow_{\mathsf{tr}} \langle t_{i-1}, R(j_{i-1}) \rangle \quad C(j_i) \Rightarrow_{\mathsf{tr}} \sigma}{c \Rightarrow_{\mathsf{tr}} c t_1 \cdots t_{i-1} \sigma}$$

$$\text{(APP-TR)}\ \frac{e_1 \Rightarrow_{\mathsf{tr}} \langle t_1, \lambda x.e\rangle \quad e_2 \Rightarrow_{\mathsf{tr}} \langle t_2, v_2\rangle \quad e[v_2/x] \Rightarrow_{\mathsf{tr}} \langle t, v\rangle}{e_1\ e_2 \Rightarrow_{\mathsf{tr}} \langle (e_1\ e_2)t_1 t_2 t, v\rangle}$$

$$\text{(DIV-APP-1)}\ \frac{e_1 \Rightarrow_{\mathsf{tr}} \sigma}{e_1\ e_2 \Rightarrow_{\mathsf{tr}} (e_1\ e_2)\sigma} \qquad \text{(DIV-APP-2)}\ \frac{e_1 \Rightarrow_{\mathsf{tr}} \langle t_1, \lambda x.e\rangle \quad e_2 \Rightarrow_{\mathsf{tr}} \sigma}{e_1\ e_2 \Rightarrow_{\mathsf{tr}} (e_1\ e_2)t_1\sigma}$$

$$\text{(DIV-APP-3)}\ \frac{e_1 \Rightarrow_{\mathsf{tr}} \langle t_1, \lambda x.e\rangle \quad e_2 \Rightarrow_{\mathsf{tr}} \langle t_2, v_2\rangle \quad e[v_2/x] \Rightarrow_{\mathsf{tr}} \sigma}{e_1\ e_2 \Rightarrow_{\mathsf{tr}} (e_1\ e_2)t_1 t_2\sigma}$$

FIGURE 5.5 Trace semantics for application

Finite trace rules enrich big-step rules in $\mathcal{R}$ by finite traces, thus modelling computations converging to a final result. On the other hand, infinite trace rules handle non-termination, modelled by infinite traces: they propagate divergence, that is, if a configuration in the premises of a rule in $\mathcal{R}$ diverges, namely, it evaluates to an infinite trace, then the subsequent premises are ignored and the configuration in the conclusion diverges as well. Note that the triple $\langle C, Tr_R^C, \mathcal{R}_{\mathsf{tr}}\rangle$ is a big-step semantics according to Definition 5.1.

The standard inductive interpretation of big-step rules is not enough in this setting: it can only derive judgements of shape $c \Rightarrow_{\mathsf{tr}} \langle t, r\rangle$, because there is no axiom introducing infinite traces, hence they cannot be derived by finite derivations. In other words, the inductive interpretation of $\mathcal{R}_{\mathsf{tr}}$ can only capture converging computations. To properly handle divergence, we have to interpret rules *coinductively*, namely, allowing both finite and infinite derivations. As usual (cf. Definition 2.6), we write $\mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} r_{\mathsf{tr}}$ to say that $c \Rightarrow_{\mathsf{tr}} r_{\mathsf{tr}}$ is coinductively derivable by rules in $\mathcal{R}_{\mathsf{tr}}$. It is important to note the following proposition, stating that enabling infinite derivations does not affect the semantics of converging computations.

LEMMA 5.16 : $\mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} \langle t, r\rangle$ iff $\mathcal{R}_{\mathsf{tr}} \vdash_\mu c \Rightarrow_{\mathsf{tr}} \langle t, r\rangle$.

Proof: The right-to-left implication is trivial, because the inductive interpretation is always included in the coinductive one. The proof of the other direction is by induction on the length of t, which is a finite trace. By hypothesis, we know that $c \Rightarrow_{\mathsf{tr}} \langle t, r\rangle$ is derivable by a (possibly infinite) derivation and, by Definition 5.15, we know that the last applied rule ρ^{tr} has shape $\mathsf{trace}(\rho,\ t_1, \ldots, t_n)$, hence $t = ct_1 \cdots t_n$. If $|t| = 1$, then $t = c$, and so $n = 0$, that is, $\rho = \mathsf{rule}(\varepsilon,\ c,\ r)$, because only non-empty traces are derivable, hence $\mathcal{R}_{\mathsf{tr}} \vdash_\mu c \Rightarrow_{\mathsf{tr}} \langle t, r\rangle$ holds by ρ^{tr}. If $|t| > 0$, then, for all $i \in 1..n$, $|t_i| < |t|$, hence, by induction hypothesis, we get $\mathcal{R}_{\mathsf{tr}} \vdash_\mu C(\rho, i) \Rightarrow_{\mathsf{tr}} \langle t_i, R(\rho, i)\rangle$, and so $\mathcal{R}_{\mathsf{tr}} \vdash_\mu c \Rightarrow_{\mathsf{tr}} \langle t, r\rangle$ holds by ρ^{tr}. □

We show in Figure 5.5 the rules obtained by applying Definition 5.15, starting from meta-rule (APP) of the example in Figure 5.1 (for the other meta-rules the outcome is analogous).

For instance, set $\Omega = \omega\,\omega$ with $\omega = \lambda x.x\,x$, and σ_Ω the infinite trace $\Omega\omega\omega\Omega\omega\omega\ldots$, it is easy to see that the judgment $\Omega \Rightarrow_{\mathsf{tr}} t_\Omega$ can be derived

by the following infinite derivation:[4]

$$\dfrac{\dfrac{}{\omega \Rightarrow_{\mathsf{tr}} \langle \omega, \omega \rangle} \quad \dfrac{}{\omega \Rightarrow_{\mathsf{tr}} \langle \omega, \omega \rangle} \quad \dfrac{\vdots}{\Omega = (x\,x)[\omega/x] \Rightarrow_{\mathsf{tr}} \sigma_\Omega}}{\Omega \Rightarrow \Omega\omega\omega\sigma_\Omega = \sigma_\Omega}$$

Note that *only* the judgment $\Omega \Rightarrow_{\mathsf{tr}} \sigma_\Omega$ can be derived, that is, the trace semantics of Ω is uniquely determined to be σ_Ω, since the infinite derivation forces the equation $\sigma_\Omega = \Omega\omega\omega\sigma_\Omega$.

To check that the construction in Definition 5.15 is a correct extension of the given big-step semantics, we have to show it is *conservative*, in the sense that it does not affect the semantics of converging computations, as formally stated below.

THEOREM 5.17 : $\mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} \langle t, r \rangle$ for some $t \in C^\star$ iff $\mathcal{R} \vdash_\mu c \Rightarrow r$.

Proof: By Lemma 5.16, we know that $\mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} \langle t, r \rangle$ iff $\mathcal{R}_{\mathsf{tr}} \vdash_\mu c \Rightarrow_{\mathsf{tr}} \langle t, r \rangle$. Then, the thesis follows by proving $\mathcal{R}_{\mathsf{tr}} \vdash_\mu c \Rightarrow_{\mathsf{tr}} \langle t, r \rangle$, for some $t \in C^\star$, iff $\mathcal{R} \vdash_\mu c \Rightarrow r$, by a straightforward induction on rules. □

We conclude this subsection by showing a coinductive proof principle associated with trace semantics, which allows us to prove that a predicate on configurations ensures the existence of a non-terminating computation.

LEMMA 5.18 : Let $\mathcal{S} \subseteq C$ be a set. If, for all $c \in \mathcal{S}$, there are $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r)$ and $i \in 1..n$ such that

1. for all $k < i$, $\mathcal{R} \vdash_\mu j_k$, and
2. $C(j_i) \in \mathcal{S}$

then, for all $c \in \mathcal{S}$, there exists $\sigma \in C^\omega$ such that $\mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} \sigma$.

Proof: First of all, for each $c \in \mathcal{S}$, we construct a trace $\sigma_c \in C^\omega$, which will be the candidate trace to prove the thesis. By hypothesis (Item 1), there is a rule $\rho_c = \mathsf{rule}(j_1^c \ldots j_{n_c}^c,\ c,\ r_c)$ and an index $i_c \in 1..n_c$ such that, for all $k < i_c$, we have $\mathcal{R} \vdash_\mu j_k^c$. Therefore, by Theorem 5.17, there are finite traces $t_1^c, \ldots, t_{i_c-1}^c \in C^\star$ such that for all $k < i_c$ we have $\mathcal{R}_{\mathsf{tr}} \vdash_\nu C(j_k^c) \Rightarrow_{\mathsf{tr}} \langle t_k^c, R(j_k^c) \rangle$, and, in addition (Item 2), we know that $C(j_{i_c}^c) \in \mathcal{S}$. Then, for each $c \in \mathcal{S}$, we can introduce a variable X_c and define an equation $X_c = c \cdot t_1^c \cdot \cdots \cdot t_{i_c-1}^c \cdot X_{C(j_{i_c}^c)}$. The set of all such equations is a guarded system of equations, which thus has a unique solution, namely, a function $s : \mathcal{S} \to C^\omega$ such that, for each $c \in \mathcal{S}$ we have $s(c) = c \cdot t_1^c \cdot \cdots \cdot t_{i_c-1}^c \cdot s(C(j_{i_c}^c))$.[a]

We now have to prove that, for all $c \in \mathcal{S}$, we have $\mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} s(c)$. To this end, consider the set $\mathcal{S}' = \{\langle c, s(c) \rangle \mid c \in \mathcal{S}\} \cup \{\langle c, \langle t, r \rangle \rangle \mid \mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} \langle t, r \rangle\}$, then the proof is by coinduction. Let $\langle c, r_{\mathsf{tr}} \rangle \in \mathcal{S}'$, then we have to find a rule $\langle j_1 \ldots j_n, c \Rightarrow_{\mathsf{tr}} r_{\mathsf{tr}} \rangle \in \mathcal{R}_{\mathsf{tr}}$ such that, for all $k \in 1..n$,

4 To help the reader, we add equivalent expressions with a grey background.

$\langle C(j_k), Tr^C_R(j_k)\rangle \in \mathcal{S}'$. We have two cases:

- if $r_{\mathsf{tr}} = s(c)$, then the needed rule is $\mathsf{trace}_\infty(\rho_c,\ i_c,\ t^c_1, \ldots, t^c_{i_c-1},\ s(C(j^c_{i_c})))$, and
- if $r_{\mathsf{tr}} = \langle t, r\rangle$, then $\mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} \langle t, r\rangle$, by construction of $\mathcal{S}'$, hence $c \Rightarrow_{\mathsf{tr}} \langle t, r\rangle$ is the conclusion of a finite trace rule, where all premises are still derivable, thus in $\mathcal{S}'$ by construction.

□

a This argument can be made more precise using coalgebras (Rutten, 2000), in particular the fact that $\mathcal{S}$ and C^ω carry, respectively, a coalgebra and a corecursive algebra (Capretta, Uustalu, and Vene, 2009) structure for the functor $X \mapsto C^\star \times X$.

5.3.2 Wrong

A well-known technique (Abadi and Cardelli, 1996; Pierce, 2002) to distinguish between stuck and diverging computations, in a sense "dual" to the previous one, is to add a special result wrong, so that $c \Rightarrow \mathsf{wrong}$ means that the evaluation of c goes stuck.

In this case, defining a general and "automatic" version of the construction, starting from an arbitrary big-step semantics $\langle C, R, \mathcal{R}\rangle$, is a non-trivial problem. Our solution is based on the equivalence on rules defined in Definition 5.12 (equality up to an index), which allows us to define when wrong can be introduced.

The extended judgement has shape $c \Rightarrow r_{\mathsf{wr}}$ where $r_{\mathsf{wr}} \in R_{\mathsf{wr}} = R + \{\mathsf{wrong}\}$, that is, it is either a result or an error. To define the extended semantics, we construct, starting from $\mathcal{R}$, an extended set of rules $\mathcal{R}_{\mathsf{wr}}$ as follows:

DEFINITION 5.19 (Rules for wrong): The set of rules $\mathcal{R}_{\mathsf{wr}}$ is obtained by adding to $\mathcal{R}$ the following rules:

WRONG CONFIGURATION RULES For each configuration $c \in C$ such that there is no rule ρ in $\mathcal{R}$ with $C(\rho) = c$, define rule $\mathsf{wrong}(c)$ as $\dfrac{}{c \Rightarrow \mathsf{wrong}}$.

WRONG RESULT RULES For each rule $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r)$ in $\mathcal{R}$, index $i \in 1..n$, and result $r' \in R$, if, for all rules ρ' such that $\rho \sim_i \rho'$, $R(\rho', i) \neq r'$, then define rule $\mathsf{wrong}(\rho,\ i,\ r')$ as

$$\frac{j_1 \quad \ldots \quad j_{i-1} \quad C(j_i) \Rightarrow r'}{c \Rightarrow \mathsf{wrong}}$$

WRONG PROPAGATION RULES These rules propagate wrong analogously to those for divergence propagation: For each rule $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r)$ in $\mathcal{R}$ and index $i \in 1..n$, define rule $\mathsf{prop}(\rho, i, \mathsf{wrong})$ as

$$\frac{j_1 \quad \ldots \quad j_{i-1} \quad C(j_i) \Rightarrow \mathsf{wrong}}{c \Rightarrow \mathsf{wrong}}$$

Wrong configurations rules simply say that, if there is no rule for a given configuration, then we can derive wrong. Wrong result rules, instead, derive

$$\text{(WRONG-APP)}\ \frac{e_1 \Rightarrow n}{e_1\ e_2 \Rightarrow \mathsf{wrong}} \qquad \text{(WRONG-SUCC)}\ \frac{e \Rightarrow \lambda x.e}{\mathsf{succ}\ e \Rightarrow \mathsf{wrong}}$$

$$\text{(PROP-APP-1)}\ \frac{e_1 \Rightarrow \mathsf{wrong}}{e_1\ e_2 \Rightarrow \mathsf{wrong}} \qquad \text{(PROP-APP-2)}\ \frac{e_1 \Rightarrow \lambda x.e \quad e_2 \Rightarrow \mathsf{wrong}}{e_1\ e_2 \Rightarrow \mathsf{wrong}}$$

$$\text{(PROP-APP-3)}\ \frac{e_1 \Rightarrow \lambda x.e \quad e_2 \Rightarrow v_2 \quad e[v_2/x] \Rightarrow \mathsf{wrong}}{e_1\ e_2 \Rightarrow \mathsf{wrong}} \qquad \text{(PROP-SUCC)}\ \frac{e \Rightarrow \mathsf{wrong}}{\mathsf{succ}\ e \Rightarrow \mathsf{wrong}}$$

FIGURE 5.6 Semantics with wrong for application and successor

wrong whenever the configuration in a premise of a rule reduces to a result which is not admitted in such (and any equivalent) rule. We also call these two kinds of rules wrong introduction rules, as they introduce wrong in the conclusion without having it in the premises. Finally, wrong propagation rules say that, if a configuration in a premise of some rule in $\mathcal{R}$ goes wrong, then the subsequent premises are ignored and the configuration in the conclusion goes wrong as well. Note that $\langle C, R_{\mathsf{wr}}, \mathcal{R}_{\mathsf{wr}} \rangle$ is a big-step semantics according to Definition 5.1.

In this case, the standard inductive interpretation is enough to get the correct semantics, because, intuitively, an error, if any, occurs after a finite number of steps. Then, we write $\mathcal{R}_{\mathsf{wr}} \vdash_\mu c \Rightarrow r_{\mathsf{wr}}$ when the judgment $c \Rightarrow r_{\mathsf{wr}}$ is inductively derivable by rules in $\mathcal{R}_{\mathsf{wr}}$.

We show in Figure 5.6 the meta-rules for wrong introduction and propagation constructed starting from those for application and successor in Figure 5.1.

For instance, rule (WRONG-APP) is introduced since in the original semantics there is rule (APP) with $e_1\ e_2$ in the conclusion and e_1 in the first premise, but there is no equivalent rule (that is, with $e_1\ e_2$ in the conclusion and e_1 in the first premise) such that the result in the first premise is n. Intuitively, this means that n is a wrong result for the evaluation of the first argument of an application.

Like the previous construction, the wrong construction is a correct extension of $\mathcal{R}$, namely, it is conservative.

THEOREM 5.20 : $\mathcal{R}_{\mathsf{wr}} \vdash_\mu c \Rightarrow r$ iff $\mathcal{R} \vdash_\mu c \Rightarrow r$.

Proof: The right-to-left implication is trivial, as $\mathcal{R} \subseteq \mathcal{R}_{\mathsf{wr}}$ by Definition 5.19. The proof of the other direction is by induction on rules in $\mathcal{R}_{\mathsf{wr}}$. The only relevant cases are rules in $\mathcal{R}$, because rules in $\mathcal{R}_{\mathsf{wr}} \setminus \mathcal{R}$ allow only the derivation of judgements of shape $c \Rightarrow \mathsf{wrong}$. Hence, the thesis is immediate. □

5.3.3 Correctness of constructions

We now prove correctness of the trace and wrong constructions, by showing they capture diverging and stuck computations, respectively, as defined by the transition relation $\longrightarrow_{\mathcal{R}}$ introduced in Section 5.2.2. This provides us a coherence result for our approach.

First of all, note that both constructions correctly capture converging computations, because, if restricted to such computations, by Theorems 5.17 and 5.20, the constructions are both equivalent to the original big-step semantics. Hence, in the following, we focus only on diverging and stuck computations, respectively.

CORRECTNESS OF $\mathcal{R}_{\mathsf{tr}}$ Given a partial evaluation tree τ, we write $\tau \longrightarrow_{\mathcal{R}}^{\omega}$ meaning that there is an infinite sequence of $\longrightarrow_{\mathcal{R}}$-steps starting from τ. Then, the theorem we want to prove is the following:

THEOREM 5.21 : $\mathcal{R}_{\mathsf{tr}} \vdash_{\nu} c \Rightarrow_{\mathsf{tr}} \sigma$, for some $\sigma \in C^{\omega}$, iff $\dfrac{}{c \Rightarrow ?} \longrightarrow_{\mathcal{R}}^{\omega}$.

To prove this result, we need to relate evaluation trees (a.k.a. derivations) in $\mathcal{R}_{\mathsf{tr}}$ to partial evaluation trees. To this end, we define a function $u_? : Tr_R^C \to R_?$, which essentially forgets traces, as follows: $u_?(\langle t, r\rangle) = r$ and $u_?(\sigma) = ?$. We can extend this function to judgements, mapping $c \Rightarrow_{\mathsf{tr}} r_{\mathsf{tr}}$ to $c \Rightarrow_{\mathsf{tr}} u_?(r_{\mathsf{tr}})$, and to rules, mapping $\mathsf{trace}(\rho,\ t_1, \ldots, t_n)$ to ρ and $\mathsf{trace}_\infty(\rho,\ i,\ t_1, \ldots, t_{i-1},\ \sigma)$ to $\mathsf{pev}_?(\rho, i, ?)$. Finally, we get a function erase that transforms an evaluation tree τ^{tr} in $\mathcal{R}_{\mathsf{tr}}$ into a partial evaluation tree, defined by $\mathsf{erase}(\tau^{\mathsf{tr}}) = u_? \circ \tau^{\mathsf{tr}}$, that is, we apply $u_?$ to all judgements labeling a node in τ^{tr}, thus erasing traces. Since $u_?$ transforms rules in $\mathcal{R}_{\mathsf{tr}}$ into rules in $\mathcal{R}_?$, the function erase is correct and satisfies the following equations between (decorated) trees.

$$\mathsf{erase}\left(_{(\mathsf{trace}(\rho,\ t_1, \ldots, t_n))}\frac{\tau_1^{\mathsf{tr}} \quad \ldots \quad \tau_n^{\mathsf{tr}}}{c \Rightarrow_{\mathsf{tr}} \langle t, r\rangle}\right) = {}_{(\rho)}\frac{\mathsf{erase}\left(\tau_1^{\mathsf{tr}}\right) \quad \ldots \quad \mathsf{erase}\left(\tau_n^{\mathsf{tr}}\right)}{c \Rightarrow r}$$

$$\mathsf{erase}\left(_{(\mathsf{trace}_\infty(\rho,\ i,\ t_1, \ldots, t_{i-1},\ \sigma))}\frac{\tau_1^{\mathsf{tr}} \quad \ldots \quad \tau_i^{\mathsf{tr}}}{c \Rightarrow_{\mathsf{tr}} \sigma'}\right) = {}_{(\mathsf{pev}_?(\rho, i, ?))}\frac{\mathsf{erase}\left(\tau_1^{\mathsf{tr}}\right) \quad \ldots \quad \mathsf{erase}\left(\tau_i^{\mathsf{tr}}\right)}{c \Rightarrow ?}$$

Note that, by construction, $\mathsf{erase}(\tau)$ is infinite and well-formed iff τ^{tr} is infinite, and $\mathsf{erase}\left(\tau^{\mathsf{tr}}\right)$ is finite iff τ^{tr} is finite.

LEMMA 5.22 : If $\mathcal{R}_{\mathsf{tr}} \vdash_{\nu} c \Rightarrow_{\mathsf{tr}} r_{\mathsf{tr}}$ holds by an infinite evaluation tree τ^{tr}, then there is a sequence $(\tau_n)_{n\in\mathbb{N}}$ such that $\tau_n \longrightarrow_{\mathcal{R}} \tau_{n+1}$ for all $n \in \mathbb{N}$, $\tau_0 = \dfrac{}{c \Rightarrow ?}$, and $\bigsqcup \tau_n = \mathsf{erase}\left(\tau^{\mathsf{tr}}\right)$.

Proof: Since τ^{tr} is infinite, $\mathsf{erase}\left(\tau^{\mathsf{tr}}\right) = \tau$ is a well-formed infinite partial evaluation treee and, by Proposition 5.11 (2), there is a strictly increasing sequence $(\tau'_n)_{n\in\mathbb{N}}$ of finite partial evaluation trees such that $\bigsqcup \tau'_n = \tau$ and $\tau'_0 = \dfrac{}{c \Rightarrow ?}$. By Proposition 5.13 (2), since for all $n \in \mathbb{N}$ we have $\tau'_n \sqsubset \tau'_{n+1}$, we get $\tau'_n \longrightarrow_{\mathcal{R}}^{\star} \tau'_{n+1}$, and, since $\tau'_n \neq \tau'_{n+1}$, this sequence of steps is not empty. Hence, we can construct a sequence $(\tau_n)_{n\in\mathbb{N}}$ such that $\tau_0 = \tau'_0 = \dfrac{}{c \Rightarrow ?}$, $\tau_n \longrightarrow_{\mathcal{R}} \tau_{n+1}$ and $\bigsqcup \tau_n = \tau$, as needed. □

Proof (Theorem 5.21): $\mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} t$ for some $t \in C^\omega$ implies $\dfrac{}{c \Rightarrow ?} \longrightarrow^\omega_{\mathcal{R}}$. Since $\mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} \sigma$ holds and σ is infinite, by (a consequence of) Lemma 5.16, there is an infinite evaluation tree τ^{tr} in $\mathcal{R}_{\mathsf{tr}}$ such that $\mathsf{r}(\tau^{\mathsf{tr}}) = c \Rightarrow_{\mathsf{tr}} \sigma$. Then, by Lemma 5.22 we get the thesis.

$\dfrac{}{c \Rightarrow ?} \longrightarrow^\omega_{\mathcal{R}}$ implies $\mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} t$ for some $t \in C^\omega$. We first prove that, if c is the root configuration of an infinite well-formed partial evaluation tree, then $\mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} \sigma$, for some $\sigma \in C^\omega$. This follows from Lemma 5.18, applied to the set $\mathcal{S} \subseteq C$ defined as follows: $c \in \mathcal{S}$ iff $C(\mathsf{r}(\tau)) = c$, for some infinite well-formed partial evaluation tree τ. Let $c \in \mathcal{S}$, then $c = C(\mathsf{r}(\tau))$ and the last applied rule in τ is $\mathsf{pev}_?(\rho, i, ?)$, for some $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r)$ in $\mathcal{R}$. Then, we have $\mathcal{R} \vdash_\mu j_k$, for all $k < i$ and $C(j_i) = C(\mathsf{r}(\tau_{|_i}))$ and $\tau_{|_i}$ is an infinite well-formed partial evaluation tree. Therefore, $C(j_i) \in \mathcal{S}$, and so hypotheses of Lemma 5.18 are satisfied. Now, by definition of $\longrightarrow^\omega_{\mathcal{R}}$, there is an infinite sequence $(\tau_n)_{n \in \mathbb{N}}$ such that $\tau_0 = \dfrac{}{c \Rightarrow ?}$ and, for all $n \in \mathbb{N}$, $\tau_n \longrightarrow_{\mathcal{R}} \tau_{n+1}$, hence, by Proposition 5.13 (1), we get $\tau_n \sqsubset \tau_{n+1}$. By Proposition 5.11 (1), we have that $\tau = \bigsqcup \tau_n$ is a well-formed infinite partial evaluation tree, hence the thesis follows from what we have just proved. □

CORRECTNESS OF $\mathcal{R}_{\mathsf{wr}}$ We now show that the construcion in Section 5.3.2 correctly models stuck computation in $\longrightarrow_{\mathcal{R}}$.

The proof relies on the following lemma. We say that a (finite) partial evaluation tree τ is *irreducible* if there is no τ' such that $\tau \longrightarrow_{\mathcal{R}} \tau'$, and it is *stuck* if it is irreducible and $R_?(\mathsf{r}(\tau)) = ?$. Note that, by Proposition 5.8 (1) and Proposition 5.13 (1), a complete partial evaluation tree τ is irreducible.

LEMMA 5.23 : If τ is a stuck partial evaluation tree with $\mathsf{r}(\tau) = c \Rightarrow ?$, then $\mathcal{R}_{\mathsf{wr}} \vdash_\mu c \Rightarrow \mathsf{wrong}$ holds.

Proof: The proof is by induction on τ, splitting cases on the last applied rule. There are three cases:

- *Case:* $\mathsf{ax}_?(c)$ Since τ is stuck, by definition of $\longrightarrow_{\mathcal{R}}$ (cf. Figure 5.3 first and second clauses), there is no rule $\rho \in \mathcal{R}$ such that $C(\rho) = c$, hence $\mathcal{R}_{\mathsf{wr}} \vdash_\mu c \Rightarrow \mathsf{wrong}$ holds, by applying $\mathsf{wrong}(c)$.
- *Case:* $\mathsf{pev}_?(\rho, i, r)$ Suppose $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r')$ and $i \in 1..n$, by hypothesis, for all $k < i$, $\tau_{|_k}$ is a complete partial evaluation tree of j_k, hence we know that $\mathcal{R} \vdash_\mu j_k$ holds. Since τ is stuck, by definition of $\longrightarrow_{\mathcal{R}}$ (cf. Figure 5.3 third and fourth clauses), there is no rule $\rho' \sim_i \rho$ with $R(\rho', i) = r$, hence $\mathsf{wrong}(\rho,\ i,\ r) \in \mathcal{R}_{\mathsf{wr}}$. By Theorem 5.20 we get $\mathcal{R}_{\mathsf{wr}} \vdash_\mu j_k$, for all $k < i$, hence applying $\mathsf{wrong}(\rho,\ i,\ r)$, we get $\mathcal{R}_{\mathsf{wr}} \vdash_\mu c \Rightarrow \mathsf{wrong}$.
- *Case:* $\mathsf{pev}_?(\rho, i, ?)$ Suppose $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r')$ and $i \in 1..n$, by hypothesis, for all $k < i$, $\tau_{|_k}$ is a complete partial evaluation tree of j_k, hence we know that $\mathcal{R} \vdash_\mu j_k$ holds. Set $c_i = C(\rho, i)$, then, since τ is stuck, by definition of $\longrightarrow_{\mathcal{R}}$ (cf. Figure 5.3 last clause), the subtree $\tau_{|_i}$ is stuck as

well and $\mathsf{r}(\tau_{|_i}) = c_i \Rightarrow ?$. By Theorem 5.20, we get $\mathcal{R}_{\mathsf{wr}} \vdash_\mu j_k$, for all $k < i$, and, by induction hypothesis, we get $\mathcal{R}_{\mathsf{wr}} \vdash_\mu c_i \Rightarrow \mathsf{wrong}$, hence, applying rule $\mathsf{prop}(\rho, i, \mathsf{wrong})$, we get $\mathcal{R}_{\mathsf{wr}} \vdash_\mu c \Rightarrow \mathsf{wrong}$.

□

THEOREM 5.24 : $\mathcal{R}_{\mathsf{wr}} \vdash_\mu c \Rightarrow \mathsf{wrong}$ iff $\dfrac{}{c \Rightarrow ?} \longrightarrow^\star_{\mathcal{R}} \tau$, where τ is stuck.

Proof: $\mathcal{R}_{\mathsf{wr}} \vdash_\mu c \Rightarrow \mathsf{wrong}$ implies $\dfrac{}{c \Rightarrow ?} \longrightarrow^\star_{\mathcal{R}} \tau$ where τ is stuck. We prove that there is a stuck tree τ with $\mathsf{r}(\tau) = c \Rightarrow ?$, then the thesis follows immediately from Proposition 5.13 (2), as we will trivially have $\dfrac{}{c \Rightarrow ?} \sqsubseteq \tau$. The proof is by induction on rules in $\mathcal{R}_{\mathsf{wr}}$. It is enough to consider only rules with wrong in the conclusion, hence we have the following three cases:

Case: $\mathit{wrong}(c)$ By Definition 5.19, there is no rule $\rho \in \mathcal{R}$ such that $C(\rho) = c$, hence $\dfrac{}{c \Rightarrow ?}$ is stuck.

Case: $\mathit{wrong}(\rho,\ i,\ r)$ By Definition 5.19, assuming $\rho \equiv \mathsf{rule}(j_1 \ldots j_n,\ c,\ r')$, there is no rule $\rho' \sim_i \rho$ such that $R(\rho', i) = r$; then, by Theorem 5.20, for all $k \leq i$, $\mathcal{R} \vdash_\mu j_k$ holds, hence there is a finite and complete partial evaluation tree τ_k with $\mathsf{r}(\tau_k) = j_k$. Therefore, applying rule $\mathsf{pev}_?(\rho, i, r)$ to $\tau_1, \ldots, \tau_i$, we get a partial evaluation tree, which is stuck, by definition of $\longrightarrow_{\mathcal{R}}$.

Case: $\mathsf{prop}(\rho, i, \mathit{wrong})$ Suppose $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r)$ and $c_i = C(j_i)$, then, by induction hypothesis, we get that there is a stuck tree τ' such that $\mathsf{r}(\tau') = c_i \Rightarrow ?$; then, by Theorem 5.20, for all $k < i$, $\mathcal{R} \vdash_\mu j_k$ holds, hence there is a finite and complete partial evaluation tree τ_k with $\mathsf{r}(\tau_k) = j_k$. Therefore, applying $\mathsf{pev}_?(\rho, i, ?)$ to $\tau_1, \ldots, \tau_{i-1}, \tau'$, we get a stuck tree.

$\dfrac{}{c \Rightarrow ?} \longrightarrow^\star_{\mathcal{R}} \tau$ where τ is stuck implies $\mathcal{R}_{\mathsf{wr}} \vdash_\mu c \Rightarrow \mathsf{wrong}$. It follows immediately from Lemma 5.23, since $\mathsf{r}(\tau) = c \Rightarrow ?$ by hypothesis. □

5.4 Divergence by coaxioms

As we have described in Section 5.3.1, traces allow us to explicitly model divergence, provided that we interpret rules coinductively: a configuration diverges if it evaluates to an infinite trace. However, the resulting semantics is somewhat redundant: traces keep track of all configurations visited during the evaluation, while we are just interested in whether there is a final result or non-termination, and a configuration may evaluate to many different infinite traces, hence divergence is modelled in many ways. In this section we show how coaxioms can be succesfully adopted to achieve a more abstract model of divergence, removing this redundancy.

The key idea is to regard divergence just as a special result ∞, that, like infinite traces (cf. Definition 5.15) and wrong (cf. Section 5.3.2), can only be

$$\text{(DIV-APP-1)}\ \frac{e_1 \Rightarrow \infty}{e_1\ e_2 \Rightarrow \infty} \qquad \text{(DIV-APP-2)}\ \frac{e_1 \Rightarrow \lambda x.e \quad e_2 \Rightarrow \infty}{e_1\ e_2 \Rightarrow \infty}$$

$$\text{(DIV-APP-3)}\ \frac{e_1 \Rightarrow \lambda x.e \quad e_2 \Rightarrow v_2 \quad e[v_2/x] \Rightarrow \infty}{e_1\ e_2 \Rightarrow \infty}$$

FIGURE 5.7 Divergence propagation rules for application

propagated by big-step rules. To this end, we define yet another construction, extending a given big-step semantics.

Let us assume a big-step semantics $\langle C, R, \mathcal{R} \rangle$. Then, the extended judgement has shape $c \Rightarrow r_\infty$ where $r_\infty \in R_\infty = R + \{\infty\}$, that is, it is either a result or divergence. To define the extended semantics, we construct, starting from $\mathcal{R}$, a new set of rules $\mathcal{R}_\infty$ as follows:

DEFINITION 5.25 (Rules for divergence): The set of rules $\mathcal{R}_\infty$ is obtained by adding to $\mathcal{R}$ the following rules:

DIVERGENCE PROPAGATION RULES For each rule $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r)$ in $\mathcal{R}$ and index $i \in 1..n$, define rule $\mathsf{prop}(\rho, i, \infty)$ as

$$\frac{j_1 \quad \ldots \quad j_{i-1} \quad C(j_i) \Rightarrow \infty}{c \Rightarrow \infty}$$

These additional rules propagate divergence, that is, if a configuration in the premises of a rule in $\mathcal{R}$ diverges, then the subsequent premises are ignored and the configuration in the conclusion diverges as well. This is very similar to infinite trace rules, but here we do not need to construct traces to represent divergence. Note that the triple $\langle C, R_\infty, \mathcal{R}_\infty \rangle$ is a big-step semantics according to Definition 5.1.

Now the question is: how do we interpret such rules? The standard inductive interpretation of big-step rules, as for trace semantics, is not enough in this setting, since there is no axiom introducing ∞, hence it cannot be derived by finite derivations. In other words, the inductive interpretation of $\mathcal{R}_\infty$ can only capture converging computations, hence it is equivalent to the inductive interpretation of $\mathcal{R}$. On the other hand, differently from trace semantics, even the coinductive interpretation cannot provide the expected semantics: it allows the derivation of too many judgements. For instance, in Figure 5.7, we report the divergence propagation rules obtained starting from meta-rule (APP) of the example in Figure 5.1 (for other meta-rules the outcome is analogous); then, using these rules (and the original ones in Figure 5.1), we can build the following infinite derivation for Ω, which is correct for any $r_\infty \in R_\infty$.

$$\frac{\dfrac{}{\omega \Rightarrow_{\mathsf{tr}} \omega} \quad \dfrac{}{\omega \Rightarrow_{\mathsf{tr}} \omega} \quad \dfrac{\vdots}{\Omega = (x\,x)[\omega/x] \Rightarrow_{\mathsf{tr}} r_\infty}}{\Omega \Rightarrow r_\infty}$$

Intuitively, we would like to allow infinite derivations only to derive divergence, namely, judgments of shape $c \Rightarrow \infty$. Inference systems with corules are

precisely the tool enabling this kind of refinement. That is, in addition to divergence propagation rules, we can add appropriate corules $\mathcal{R}_{co}$ for divergence, as defined below.

DEFINITION 5.26 (Coaxioms for divergence): The set of corules $\mathcal{R}_{co}$ consists of the following coaxioms:

COAXIOMS FOR DIVERGENCE for each configuration $c \in C$, define coaxiom $\mathsf{div}_{co}(c)$ as $\dfrac{}{c \Rightarrow \infty}$.

As described in detail in Chapter 3, coaxioms impose additional conditions on infinite derivations to be considered correct: a judgement $c \Rightarrow r_\infty$ is derivable in $\langle \mathcal{R}_\infty, \mathcal{R}_{co} \rangle$ iff it has an arbitrary (finite or infinite) derivation in $\mathcal{R}_\infty$, whose nodes all have a finite derivation in $\mathcal{R}_\infty \cup \mathcal{R}_{co}$, that is, using both rules and corules. We will write $\langle \mathcal{R}_\infty, \mathcal{R}_{co} \rangle \vdash_\nu c \Rightarrow r_\infty$ when $c \Rightarrow r_\infty$ is derivable in $\langle \mathcal{R}_\infty, \mathcal{R}_{co} \rangle$.

In the above example, $\langle \mathcal{R}_\infty, \mathcal{R}_{co} \rangle \vdash_\nu \Omega \Rightarrow r_\infty$ holds iff $r_\infty = \infty$, because $\Omega \Rightarrow r$ has no finite derivation in $\mathcal{R}_\infty \cup \mathcal{R}_{co}$, for any $r \in R$. In the case of the trace construction (cf. Section 5.3.1), coaxioms are not needed as rules are *productive*, because the trace in the conclusion is always strictly larger than those in the premises, see Definition 5.15.

To check that the construction in Definition 5.25 and Definition 5.26 is a correct extension of the given big-step semantics, as for trace semantics, we have to show it is conservative, in the sense that it does not affect the semantics of converging computations, as formally stated below.

THEOREM 5.27 : $\langle \mathcal{R}_\infty, \mathcal{R}_{co} \rangle \vdash_\nu c \Rightarrow r$ iff $\mathcal{R} \vdash_\mu c \Rightarrow r$.

Proof: The right-to-left implication is trivial as $\mathcal{R} \subseteq \mathcal{R}_\infty$ by Definition 5.25. To get the other direction, note that if $\langle \mathcal{R}_\infty, \mathcal{R}_{co} \rangle \vdash_\nu c \Rightarrow r$ then we have $\mathcal{R}_\infty \cup \mathcal{R}_{co} \vdash_\mu c \Rightarrow r$. Hence, we prove by induction on rules in $\mathcal{R}_\infty \cup \mathcal{R}_{co}$ that, if $\mathcal{R}_\infty \cup \mathcal{R}_{co} \vdash_\mu c \Rightarrow r$ then $\mathcal{R} \vdash_\mu c \Rightarrow r$. The cases of coaxiom $\mathsf{div}_{co}(c)$ and divergence propagation $\mathsf{prop}(\rho, i, \infty)$ are both empty, as the conclusion of such rules has shape $c \Rightarrow \infty$. The only relevant case is that of a rule $\rho \in \mathcal{R}$, for which the thesis follows immediately. □

Inference systems with corules come with the bounded coinduction principle. Thanks to such principle, we can define a coinductive proof principle, which allows us to prove that a predicate on configurations ensures the existence of a non-terminating computation.

LEMMA 5.28 : Let $\mathcal{S} \subseteq C$ be a set. If, for all $c \in \mathcal{S}$, there are $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r)$ in $\mathcal{R}$ and $i \in 1..n$ such that

1. for all $k < i$, $\mathcal{R} \vdash_\mu j_k$, and
2. $C(j_i) \in \mathcal{S}$

then, for all $c \in \mathcal{S}$, $\langle \mathcal{R}_\infty, \mathcal{R}_{co} \rangle \vdash_\nu c \Rightarrow \infty$.

Proof: Consider the set $\mathcal{S}' = \{\langle c, \infty\rangle \mid c \in \mathcal{S}\} \cup \{\langle c, r\rangle \mid \mathcal{R} \vdash_\mu c \Rightarrow r\}$, then the proof is by bounded coinduction (cf. Proposition 3.27).

BOUNDEDNESS We have to show that, for all $\langle c, r_\infty\rangle \in \mathcal{S}'$, $\mathcal{R}_\infty \cup \mathcal{R}_{\mathsf{co}} \vdash_\mu c \Rightarrow r_\infty$ holds. This is easy because, if $r_\infty = \infty$, then this holds by coaxiom $\mathsf{div}_{\mathsf{co}}(c)$, otherwise $r_\infty \in R$ and $\mathcal{R} \vdash_\mu c \Rightarrow r_\infty$, hence this holds since $\mathcal{R} \subseteq \mathcal{R}_\infty \subseteq \mathcal{R}_\infty \cup \mathcal{R}_{\mathsf{co}}$.

CONSISTENCY We have to show that, for all $\langle c, r_\infty\rangle \in \mathcal{S}'$, there is a rule $\langle j_1 \dots j_n, c \Rightarrow r_\infty\rangle \in \mathcal{R}_\infty$ such that, for all $k \in 1..n$, $\langle C(j_k), R_\infty(j_k)\rangle \in \mathcal{S}'$. There are two cases:

- If $r_\infty = \infty$, then by hypothesis (Item 1), we have a rule $\rho = \mathsf{rule}(j_1 \dots j_n,\ c,\ r) \in \mathcal{R}$ and an index $i \in 1..n$ such that, for all $k < i$, $\mathcal{R} \vdash_\mu j_k$ and $C(j_i) \in \mathcal{S}$. Then, the needed rule is $\mathsf{prop}(\rho, i, \infty)$.
- If $r_\infty \in R$, then, by construction of $\mathcal{S}'$, we have $\mathcal{R} \vdash_\mu c \Rightarrow r_\infty$, hence, there is a rule $\rho = \mathsf{rule}(j_1 \dots j_n,\ c,\ r_\infty) \in \mathcal{R} \subseteq \mathcal{R}_\infty$, where, for all $k \in 1..n$, $\mathcal{R} \vdash_\mu j_k$ holds, and so $\langle C(j_k), R(j_k)\rangle \in \mathcal{S}'$.

□

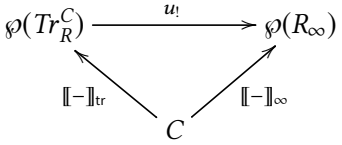
The reader may have noticed that most definitions and results in this section are very similar to those provided for trace semantics in Section 5.3.1. This is not a coincidence, indeed, we now formally prove this semantics is an *abstraction* of trace semantics.

Intuitively, if we are only interested in modelling convergence or divergence, traces are useless, in the sense that it is only relevent to know whether the trace is infinite or not and, in case it is finite, the final result. We can model this intuition by a (surjective) function $u : Tr_R^C \to R_\infty$ simply forgetting traces, that is, $u(\langle t, r\rangle) = r$ and $u(\sigma) = \infty$, with $t \in C^\star$ and $\sigma \in C^\omega$.

Then, we aim at proving the following result:

THEOREM 5.29 : $\langle \mathcal{R}_\infty, \mathcal{R}_{\mathsf{co}}\rangle \vdash_\nu c \Rightarrow r_\infty$ iff $\mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} r_{\mathsf{tr}}$, for some r_{tr} such that $r_\infty = u(r_{\mathsf{tr}})$.

In a diagrammatic form, Theorem 5.29 says that the following diagram commutes:



where $u_! : \wp(\mathcal{R}_{\mathsf{tr}}) \to \wp(\mathcal{R}_\infty)$ is the direct image of u, $[\![-]\!]_{\mathsf{tr}} : C \to \wp(Tr_R^C)$ is defined by $[\![c]\!]_{\mathsf{tr}} = \{r_{\mathsf{tr}} \in Tr_R^C \mid \mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} r_{\mathsf{tr}}\}$, and $[\![-]\!]_\infty : C \to \wp(R_\infty)$ is defined by $[\![c]\!]_{\mathsf{tr}} = \{r_\infty \in R_\infty \mid \langle \mathcal{R}_\infty, \mathcal{R}_{\mathsf{co}}\rangle \vdash_\nu c \Rightarrow r_\infty\}$.

Proof: The statement can be split in the following two points:

1. $\langle \mathcal{R}_\infty, \mathcal{R}_{\mathsf{co}}\rangle \vdash_\nu c \Rightarrow r$ iff $\mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} \langle t, r\rangle$, for some $t \in C^\star$, and

2. $\langle \mathcal{R}_\infty, \mathcal{R}_{\mathsf{co}} \rangle \vdash_\nu c \Rightarrow \infty$ iff $\mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} \sigma$, for some $\sigma \in C^\omega$.

The first point follows immediately from Theorem 5.17 and Theorem 5.27, as $\langle \mathcal{R}_\infty, \mathcal{R}_{\mathsf{co}} \rangle \vdash_\nu c \Rightarrow r$ and $\mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} \langle t, r \rangle$ are both equivalent to $\mathcal{R} \vdash_\mu c \Rightarrow r$. Then, we have only to prove the second point.

The left-to-right implication follows applying Lemma 5.18 to the set $\mathcal{S}_\infty = \{c \in C \mid \langle \mathcal{R}_\infty, \mathcal{R}_{\mathsf{co}} \rangle \vdash_\nu c \Rightarrow \infty\}$. If $c \in \mathcal{S}_\infty$, then $c \Rightarrow \infty$ is derived by a rule $\mathsf{prop}(\rho, i, \infty)$ for some $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r)$ in $\mathcal{R}$ and $i \in 1..n$, hence we have $\langle \mathcal{R}_\infty, \mathcal{R}_{\mathsf{co}} \rangle \vdash_\nu j_k$, which implies $\mathcal{R} \vdash_\mu j_k$ by Theorem 5.27, for all $k < i$, and $\langle \mathcal{R}_\infty, \mathcal{R}_{\mathsf{co}} \rangle \vdash_\nu C(j_i) \Rightarrow \infty$, that is, $C(j_i) \in \mathcal{S}_\infty$, because these judgements are the premises of $\mathsf{prop}(\rho, i, \infty)$. Therefore, the hypotheses of Lemma 5.18 are satisfied and we get, for all $c \in \mathcal{S}_\infty$, $\mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} \sigma_c$, for some $\sigma_c \in C^\omega$, hence $u(\sigma_c) = \infty$.

Similarly, the right-to-left implication follows applying Lemma 5.28 to the set $\mathcal{S}_{\mathsf{tr}} = \{c \in C \mid \mathcal{R}_{\mathsf{tr}} \vdash_\nu c \Rightarrow_{\mathsf{tr}} \sigma \text{ for some } \sigma \in C^\omega\}$. If $c \in \mathcal{S}_{\mathsf{tr}}$, then, for some $\sigma \in C^\omega$, $c \Rightarrow_{\mathsf{tr}} \sigma$ is derived by a rule $\mathsf{trace}_\infty(\rho,\ i,\ t_1, \ldots, t_{i-1},\ \sigma')$, for some $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r)$ in $\mathcal{R}$ and $i \in 1..n$, hence we have $\mathcal{R}_{\mathsf{tr}} \vdash_\nu C(j_k) \Rightarrow_{\mathsf{tr}} \langle t_k, R(j_k) \rangle$, which implies $\mathcal{R} \vdash_\mu j_k$ by Theorem 5.17, for all $k < i$, and $\mathcal{R}_{\mathsf{tr}} \vdash_\nu C(j_i) \Rightarrow_{\mathsf{tr}} \sigma'$, that is, $C(j_i) \in \mathcal{S}_{\mathsf{tr}}$, because these judgements are the premises of the rule $\mathsf{trace}_\infty(\rho,\ i,\ t_1, \ldots, t_{i-1},\ \sigma')$. Therefore, the hypotheses of Lemma 5.28 are satisfied and we get, for all $c \in \mathcal{S}_{\mathsf{tr}}$, $\langle \mathcal{R}_\infty, \mathcal{R}_{\mathsf{co}} \rangle \vdash_\nu c \Rightarrow \infty$. □

As an immediate consequence of Theorem 5.29 and Theorem 5.21, we get the following corollary, stating that the costruction given by Definitions 5.25 and 5.26 correctly models diverging computations:

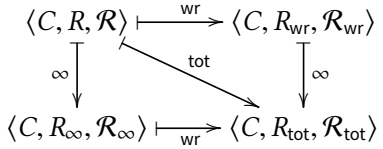
COROLLARY 5.30 : $\langle \mathcal{R}_\infty, \mathcal{R}_{\mathsf{co}} \rangle \vdash_\nu c \Rightarrow \infty$ iff $\dfrac{}{c \Rightarrow ?} \longrightarrow^\omega_{\mathcal{R}}$.

TOTAL SEMANTICS We now briefly describe how we can combine the presented constructions in order to get a semantics modelling *all* computations as defined in Section 5.2.2. In particular, we will use the wrong construction to model stuck computations and the construction in this section to model divergence, because they are more similar to each other.

Let us consider a big-step semantics $\langle C, R, \mathcal{R} \rangle$. We add to R two special values to model stuckness and divergence, defining $R_{\mathsf{tot}} = R + \{\mathsf{wrong}\} + \{\infty\}$. Then, we have to add appropriate rules to handle these two special results: the idea is to add "simultanously" rules from Definition 5.19 and from Definition 5.25, that is, we define $\mathcal{R}_{\mathsf{tot}} = \mathcal{R}_{\mathsf{wr}} \cup \mathcal{R}_\infty$. Note that, since both $\mathcal{R}_{\mathsf{wr}}$ and $\mathcal{R}_\infty$ extend $\mathcal{R}$, we have $\mathcal{R} \subseteq \mathcal{R}_{\mathsf{tot}}$. In addition, the triple $\langle C, R_{\mathsf{tot}}, \mathcal{R}_{\mathsf{tot}} \rangle$ is a big-step semantics according to Definition 5.1. Finally, to properly model divergence, we have to add corules from Definition 5.26, so that infinite derivations are only allowed to prove divergence.

Since, as we have noticed, all the presented constructions yield a big-step semantics, starting from another one, we can also try to combine them "sequentially". Of course, there are two possibilities: either we first apply the wrong construction or the divergence construction. Nicely, it is not difficult to check

that all these possibilities yield the same big-step semantics $\langle C, R_{\mathsf{tot}}, \mathcal{R}_{\mathsf{tot}} \rangle$, as depicted below:



Thanks to the commutativity of the above diagram, we can exploit results proved for the various constructions to get properties of this last construction, as stated below.

PROPOSITION 5.31 : The following facts hold:

1. $\langle \mathcal{R}_{\mathsf{tot}}, \mathcal{R}_{\mathsf{co}} \rangle \vdash_{\nu} c \Rightarrow r$ iff $\mathcal{R} \vdash_{\mu} c \Rightarrow r$,
2. $\langle \mathcal{R}_{\mathsf{tot}}, \mathcal{R}_{\mathsf{co}} \rangle \vdash_{\nu} c \Rightarrow \mathsf{wrong}$ iff $\mathcal{R}_{\mathsf{wr}} \vdash_{\mu} c \Rightarrow \mathsf{wrong}$,
3. $\langle \mathcal{R}_{\mathsf{tot}}, \mathcal{R}_{\mathsf{co}} \rangle \vdash_{\nu} c \Rightarrow \infty$ iff $\langle \mathcal{R}_{\infty}, \mathcal{R}_{\mathsf{co}} \rangle \vdash_{\nu} c \Rightarrow \infty$.

Proof: All right-to-left implication are trivial, as $\mathcal{R}, \mathcal{R}_{\mathsf{wr}}, \mathcal{R}_{\infty} \subseteq \mathcal{R}_{\mathsf{tot}}$. The other implications follow from Theorems 5.20 and 5.27, relying on the above commutative diagram. □

COROLLARY 5.32 : For any configuration $c \in C$, one of the following holds:

- either $\langle \mathcal{R}_{\mathsf{tot}}, \mathcal{R}_{\mathsf{co}} \rangle \vdash_{\nu} c \Rightarrow r$, for some $r \in R$,
- or $\langle \mathcal{R}_{\mathsf{tot}}, \mathcal{R}_{\mathsf{co}} \rangle \vdash_{\nu} c \Rightarrow \infty$,
- or $\langle \mathcal{R}_{\mathsf{tot}}, \mathcal{R}_{\mathsf{co}} \rangle \vdash_{\nu} c \Rightarrow \mathsf{wrong}$.

Proof: Straightforward from Proposition 5.31 and Theorems 5.21, 5.24 and 5.29, since the partial evaluation tree $\dfrac{}{c \Rightarrow ?}$, either converges to a tree, which is either complete or stuck, or diverges. □

Note that these three possibilities in general are not mutually exclusive, that is, for instance, a configuration can both converge to a result and diverge. This is due to the fact that big-step rules can define a non-deterministic behaviour.

5.5 Expressing and proving soundness

A predicate (for instance, a typing judgment) is *sound* when, informally, a program satisfying such predicate (*e.g.*, a well-typed program) cannot *go wrong*, following Robin Milner's slogan (Milner, 1978). In small-step style, as firstly formulated by Wright and Felleisen (1994), this is naturally expressed as follows: well-typed programs never reduce to terms which neither are values, nor can be further reduced (called *stuck* terms). The standard technique to ensure soundness is by subject reduction (well-typedness is preserved by reduction) and progress (a well-typed term is not stuck).

In standard (inductive) big-step semantics, soundness, as described above, cannot even be expressed, because diverging and stuck computations are not distinguishable.

Constructions presented in the previous sections make this distinction explicit, hence they allow us to reason about soundness with respect to a big-step semantics. Then, in this section, we discuss how soundness can be expressed and we will provide sufficient conditions. In other words, we provide a proof technique to show the soundness of a predicate with respect to a big-step semantics.

It is important to highlight the following about the presented approach to soundness. First, even though type systems are the paradigmatic example, we will consider a generic predicate on configurations, hence our approach could be instantiated with other kinds of predicates. Second, depending on the kind of construction considered, we can express different flavours of soundness, which will have different proof techniques. Finally, and more importantly, as mentioned in the introduction of the chapter, the extended semantics is only needed to prove the correctness of the technique, whereas to *apply* the technique for a given big-step semantics it is enough to reason on the original rules.

5.5.1 Expressing soundness

In the following, we assume a big-step semantics $\langle C, R, \mathcal{R} \rangle$, and an *indexed predicate on configurations and results*, that is, a family $\Pi = \langle \Pi_\iota^C, \Pi_\iota^R \rangle_{\iota \in I}$, for I set of *indexes*, with $\Pi_\iota^C \subseteq C$ and $\Pi_\iota^R \subseteq R$. A representative case is that, as in the examples of Section 5.6, predicates on configurations and results are typing judgments and the indexes are types; however, this setting is more general and so the proof technique could be applied to other kinds of predicates. When there is no ambiguity, we also denote by Π^C and Π^R, respectively, the corresponding predicates $\bigcup_{\iota \in I} \Pi_\iota^C$ and $\bigcup_{\iota \in I} \Pi_\iota^R$ on C and R (*e.g.*, to be well-typed with an arbitrary type).

To discuss how to express soundness of Π, first of all note that, in the non-deterministic case (that is, there is possibly more than one computation for a configuration), we can distinguish two flavours of soundness, see, *e.g.*, (De Nicola and Hennessy, 1984):

SOUNDNESS-MUST (or simply soundness) no computation can be stuck

SOUNDNESS-MAY at least one computation is not stuck

Soundness-must is the standard soundness in small-step semantics, and can be expressed by the wrong construction as follows:

SOUNDNESS-MUST If $c \in \Pi^C$, then $\mathcal{R}_{\mathsf{wr}} \not\vdash_\mu c \Rightarrow \mathsf{wrong}$

Instead, soundness-must *cannot* be expressed by the constructions making divergence explicit, because stuck computations are not explicitly modelled. In contrast, soundness-may can be expressed, for instance, by the divergence construction as follows:

SOUNDNESS-MAY If $c \in \Pi^C$, then $\langle \mathcal{R}_\infty, \mathcal{R}_{co} \rangle \vdash_\nu c \Rightarrow r_\infty$, for some $r_\infty \in R_\infty$

whereas cannot be expressed by the wrong construction, since diverging computations are not modelled. Note that, instead, using the total semantics, we can express both flavours of soundness, as it models both diverging and stuck computations.

Of course soundness-must and soundness-may coincide in the deterministic case. Finally, note that indexes (*e.g.*, the specific types of configurations and results) do not play any role in the above statements. However, they are relevant in the notion of *strong soundness*, introduced by Wright and Felleisen (1994). Strong soundness holds (in must or may flavour) if soundness holds (in must or may flavour), and, moreover, configurations satisfying Π^C_ι (*e.g.*, having a given type) produce results, if any, satisfying Π^R_ι (*e.g.*, of the same type). Note that soundness alone does not even guarantee to obtain a result satisfying Π^R (*e.g.*, a well-typed result). The sufficient conditions introduced in the following subsection actually ensure strong soundness.

In Section 5.5.2, we provide sufficient conditions for soundness-must, showing that they ensure soundness as stated above (Theorem 5.38). Then, in Section 5.5.3, we provide (weaker) sufficient conditions for soundness-may, and show that they ensure soundness-may (Theorem 5.41).

5.5.2 Conditions ensuring soundness-must

The three conditions which ensure the soundness-must property are *local preservation*, ∃*-progress*, and ∀-progress. The names suggest that the former plays the role of the *type preservation (subject reduction)* property, and the latter two of the *progress* property in small-step semantics. However, as we will see, the correspondence is only rough, since the reasoning here is different.

Considering the first condition more closely, we use the name *preservation* rather than type preservation since, as already mentioned, the proof technique can be applied to arbitrary predicates. More importantly, *local* means that the condition is *on single rules* rather than on the semantic relation as a whole, as standard subject reduction. The same holds for the other two conditions.

DEFINITION 5.33 (Local preservation (LP)): For each $\rho = \mathsf{rule}(j_1 \dots j_n,\ c,\ r)$ in $\mathcal{R}$, if $c \in \Pi^C_\iota$, then there exists $\iota_1, \dots, \iota_n \in I$ such that

1. for all $k \in 1..n$, if, for all $h < k$, $\mathcal{R} \vdash_\mu j_h$ and $R(j_h) \in \Pi^R_{\iota_h}$, then $C(j_k) \in \Pi^C_{\iota_k}$, and
2. if, for all $k \in 1..n$, $\mathcal{R} \vdash_\mu j_k$ and $R(j_k) \in \Pi^R_{\iota_k}$, then $r \in \Pi^R_\iota$.

Thinking to the paradigmatic case where the indexes are types, to check that this condition holds, for each rule $\rho = \mathsf{rule}(j_1 \dots j_n,\ c,\ r)$ where c, the conclusion, has type ι, we have to find types $\iota_1, \dots, \iota_n$, which can be assigned to (configurations and results in) the premises, and, when all the premises

satisfy the chosen type, r, the result in the conclusion, must have type ι, that is, the same type of c. More precisely, we will proceed as follows: we start finding type ι_1, and successively find the type ι_k for (the configuration in) the k-th premise assuming that all previous premises are derivable and their results have the expected types, and, finally, we have to check that the final result r has type ι assuming all premises are derivable and their results have the expected type. Indeed, if all such previous premises are derivable, then the expected type should be preserved by their results; if some premise is not derivable, the considered rule is "useless". For instance, considering (an instantiation of) meta-rule (APP) $\mathsf{rule}(e_1 \Rightarrow \lambda x.e\ e_2 \Rightarrow v_2\ e[v_2/x] \Rightarrow v,\ e_1\ e_2,\ v)$ in Figure 5.1, we prove that $e[v_2/x]$ has the type T of $e_1\ e_2$ under the assumption that $\lambda x.e$ has type $T' \to T$, and v_2 has type T' (see the proof example in Section 5.6.1 for more details). A counter-example to condition (LP) is discussed at the beginning of Section 5.6.3.

The following lemma states that local preservation actually implies *preservation* of the semantic relation as a whole.

LEMMA 5.34 (Preservation): Let $\langle C, R, \mathcal{R}\rangle$ and $\Pi = \langle \Pi_\iota^C, \Pi_\iota^R\rangle_{\iota\in I}$ satisfy condition (LP). If $\mathcal{R} \vdash_\mu c \Rightarrow r$ and $c \in \Pi_\iota^C$, then $r \in \Pi_\iota^R$.

Proof: The proof is by a double induction. We denote by *RH* and *IH* the first and the second induction hypothesis, respectively. The first induction is on big-step rules. Consider a rule $\rho = \mathsf{rule}(j_1 \dots j_n,\ c,\ r)$ with $c \in \Pi_\iota^C$. We prove by complete arithmetic induction on $k \in 1..n$ that $C(j_k) \in \Pi_{\iota_k}$, for all $k \in 1..n$ and for some $\iota_1, \dots, \iota_n \in I$. By (LP), there are indexes $\iota_1, \dots, \iota_n \in I$, satisfying Items 1 and 2 of (LP) (cf. Definition 5.33). Let $k \in 1..n$, then by *IH* we know that $C(j_h) \in \Pi_{\iota_h}^C$, for all $h < k$. Then, by *RH*, we get that $R(j_h) \in \Pi_{\iota_h}^R$. Hence, by (LP) (cf. Definition 5.33 (1)), we get $C(j_k) \in \Pi_{\iota_k}$, as needed.

Now, since $C(j_k) \in \Pi_{\iota_k}^C$, for all $k \in 1..n$, as we have just proved, again by *RH*, we get that $R(j_k) \in \Pi_{\iota_k}^R$, for all $k \in 1..n$. Then, by (LP) (cf. Definition 5.33 (2)), we conclude that $r \in \Pi_\iota^R$, as needed. □

The following proposition is a form of local preservation where indexes (*e.g.*, specific types) are not relevant, simpler to use in the proofs of Theorems 5.38 and 5.41.

PROPOSITION 5.35 : Let $\langle C, R, \mathcal{R}\rangle$ and $\Pi = \langle \Pi_\iota^C, \Pi_\iota^R\rangle_{\iota\in I}$ satisfy condition (LP). For each rule $\rho = \mathsf{rule}(j_1 \dots j_n,\ c,\ r)$ and $k \in 1..n$, if $c \in \Pi^C$ and, for all $h < k$, $\mathcal{R} \vdash_\mu j_h$, then $C(j_k) \in \Pi^C$.

Proof: By hypothesis we know that $c \in \Pi_\iota^C$, for some $\iota \in I$, thus by condition (LP), there are indexes $\iota_1, \dots, \iota_n \in I$, satisfying Items 1 and 2 of (LP) (cf. Definition 5.33). We show by complete arithmetic induction that, for all $k \in 1..n$, $C(j_k) \in \Pi_{\iota_k}^C$, which implies the thesis. Assume the thesis for all $h < k$, then, since by hypothesis we have $\mathcal{R} \vdash_\mu j_h$ for all $h < k$, we get, by induction hypothesis, $C(j_h) \in \Pi_{\iota_h}^C$, for all $h < k$. By Lemma 5.34, we also

get $R(j_h) \in \Pi^R_{\iota_h}$, hence, by condition (LP) (cf. Definition 5.33 (1)), we get $C(j_k) \in \Pi^C_{\iota_k}$, as needed. □

The second condition, named ∃*-progress*, ensures that, for configurations satisfying Π (*e.g.*, well-typed), we can *start* the evaluation, that is, the construction of an evaluation tree.

DEFINITION 5.36 (∃-progress (∃P)): For each $c \in \Pi^C$, there exists a rule $\rho \in \mathcal{R}$ such that $C(\rho) = c$.

The third condition, named ∀*-progress*, ensures that, for configurations satisfying Π (*e.g.*, well-typed), we can *continue* the evaluation, that is, the construction of the evaluation tree. This condition uses the equivalence on rules introduced in Definition 5.12.

DEFINITION 5.37 (∀-progress (∀P)): For each rule $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r)$, if $c \in \Pi^C$, then, for each $k \in 1..n$, if, for all $h < k$, $\mathcal{R} \vdash_\mu j_h$ and $\mathcal{R} \vdash_\mu C(j_k) \Rightarrow r'$, for some $r' \in R$, then there is a rule $\rho' \sim_k \rho$ such that $R(\rho', k) = r'$.

We have to check, for each rule $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r)$, the following: if the configuration c in the conclusion satisfies the predicate (*e.g.*, is well-typed), then, for each $k \in 1..n$, if the configuration in the k-th premise evaluates to some result r' (that is, $\mathcal{R} \vdash_\mu C(j_k) \Rightarrow r'$), then there is a rule (ρ itself or another rule with the same configuration in the conclusion and the first $k-1$ premises) with such judgement as k-th premise. This check can be done under the assumption that all the previous premises are derivable. For instance, consider again (an instantiation of) the meta-rule (APP) $\mathsf{rule}(e_1 \Rightarrow \lambda x.e\ e_2 \Rightarrow v_2\ e[v_2/x] \Rightarrow v,\ e_1\ e_2,\ v)$. Assuming that e_1 evaluates to some v_1, we have to check that there is a rule with first premise $e_1 \Rightarrow v_1$, in practice, that v_1 is a λ-abstraction; in general, checking (∀P) for a (meta-)rule amounts to show that configurations in the premises evaluate to results with the required shape (see also the proof example in Section 5.6.1).

We now prove the claim of soundness-must expressed by means of the wrong construction (cf. Section 5.3.2).

THEOREM 5.38 (Soundness-must): Let $\langle C, R, \mathcal{R}\rangle$ and $\Pi = \langle \Pi^C_\iota, R_\iota\rangle_{\iota \in I}$ satisfy conditions (LP), (∃P) and (∀P). If $c \in \Pi^C$, then $\mathcal{R}_{\mathsf{wr}} \not\vdash_\mu c \Rightarrow \mathsf{wrong}$.

Proof: To prove the statement, we assume $\mathcal{R}_{\mathsf{wr}} \vdash_\mu c \Rightarrow \mathsf{wrong}$ and look for a contradiction. The proof is by induction on the derivation of $c \Rightarrow \mathsf{wrong}$. We split cases on the last applied rule in such derivation.

Case: $\mathit{wrong}(c)$ By construction (cf. Definition 5.19), we know that there is no rule $\rho \in \mathcal{R}$ such that $C(\rho) = c$, and this violates condition (∃P), since $c \in \Pi^C$, by hypothesis.

Case: $\mathit{wrong}(\rho,\ i,\ r')$ Suppose $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r)$, hence $i \in 1..n$, then, by hypothesis, for all $k < i$, we have $\mathcal{R}_{\mathsf{wr}} \vdash_\mu j_k$, and $\mathcal{R}_{\mathsf{wr}} \vdash_\mu C(j_i) \Rightarrow r'$,

and these judgments can also be derived in $\mathcal{R}$ by conservativity (cf. Theorem 5.20). Furthermore, by construction (cf. Definition 5.19), we know that there is no other rule $\rho' \sim_i \rho$ such that $R(\rho', i) = r'$, and this violates condition (∀P), since $c \in \Pi^C$ by hypothesis.

Case: $\mathsf{prop}(\rho, i, \mathit{wrong})$ Suppose $\rho = \mathsf{rule}(j_1 \dots j_n,\ c,\ r)$, hence $i \in 1..n$, then, by hypothesis, for all $k < i$, we have $\mathcal{R}_{\mathsf{wr}} \vdash_\mu j_k$, and these judgments can also be derived in $\mathcal{R}$ by conservativity (cf. Theorem 5.20). Then, by Proposition 5.35 (which requires condition (LP)), since $c \in \Pi^C$, we have $C(j_i) \in \Pi^C$, hence we get the thesis by induction hypothesis, because $\mathcal{R}_{\mathsf{wr}} \vdash_\mu C(j_i) \Rightarrow \mathsf{wrong}$ holds by hypothesis.

□

Note that conditions (LP), (∃P) and (∀P), actually ensures strong soundness, because, by Lemma 5.34, which is applicable since we assume (LP), we have that converging computations preserve indexes of the predicate.

5.5.3 Conditions ensuring soundness-may

As discussed in Section 5.5.1, if we explicitly model divergence rather than stuck computations, we can only express a weaker form of soundness: at least one computation is not stuck (*soundness-may*). Actually, we will state soundness-may in a different, but equivalent, way, which is simpler to prove, that is, a configuration that does not converge, diverges.

As the reader can expect, to ensure this property weaker sufficient conditions are enough: namely, condition (LP), and another condition, named *may-progress*, defined below. We write $\mathcal{R} \nvdash_\mu c \Rightarrow$ if c *does not converge* (there is no r such that $\mathcal{R} \vdash_\mu c \Rightarrow r$).

DEFINITION 5.39 (May-progress (MAYP)): For each $c \in \Pi^C$, there is a rule $\rho = \mathsf{rule}(j_1 \dots j_n,\ c,\ r)$ such that, if there is a (first) $k \in 1..n$ such that $\mathcal{R} \nvdash_\mu j_k$ and, for all $h < k$, $\mathcal{R} \vdash_\mu j_h$, then $\mathcal{R} \nvdash_\mu C(j_k) \Rightarrow$.

This condition can be informally understood as follows: we have to show that there is an either finite or infinite computation for c. If we find a rule where all premises are derivable (there is no k), then there is a finite computation. Otherwise, c does not converge. In this case, we should find a rule where the configuration in the first non-derivable premise k does not converge as well. Indeed, by coinductive reasoning (use of Lemma 5.28), we obtain that c diverges. The following proposition states that this condition is indeed a weakening of (∃P) and (∀P).

PROPOSITION 5.40 : Conditions ∃-progress (∃P) and ∀-progress (∀P) imply condition may-progress (MAYP).

Proof: For each $c \in C$, let us define $b_c \in \mathbb{N}$ as $\max\{\#\rho \mid C(\rho) = c\}$, which is well-defined and finite by Assumption 5.1. For each rule ρ with $C(\rho) = c$, let us denote by $nd(\rho)$ the index of the first premise of ρ which is not derivable, if any, otherwise set $nd(\rho) = b_c$. For each $c \in \Pi^C$, we first prove the following fact: ($\star$) for each rule ρ, with $C(\rho) = c$, there exists a rule ρ' such that $C(\rho') = c$, $nd(\rho') \geq nd(\rho)$ and, if $nd(\rho') \leq b_c$, then, for all $r \in R, \mathcal{R} \nvdash_\mu C(\rho', nd(\rho')) \Rightarrow r$. Note that the requirement in ($\star$) is the same as that of condition (MAYP). The proof is by complete arithmetic induction on $h(\rho) = b_c + 1 - nd(\rho)$. If $h(\rho) = 0$, hence $nd(\rho) = b_c + 1$, then the thesis follows by taking $\rho' = \rho$. Otherwise, we have two cases: if there is no $r \in R$ such that $\mathcal{R} \vdash_\mu C(\rho, nd(\rho)) \Rightarrow r$, then we have the thesis taking $\rho' = \rho$; otherwise, by condition ($\forall$P), there is a rule $\rho'' \sim_{nd(\rho)} \rho$ such that $R(\rho'', nd(\rho)) = r$, hence $nd(\rho'') > nd(\rho)$. Then, we have $h(\rho'') < h(\rho)$, hence we get the thesis by induction hypothesis.

Now, by condition ($\exists$P), there is a rule ρ with $C(\rho) = c$, and applying ($\star$) to ρ we get condition (MAYP). □

We now prove the claim of soundness-may expressed by means of the divergence construction (cf. Section 5.4).

THEOREM 5.41 (Soundness-may): Let $\langle C, R, \mathcal{R}\rangle$ and $\Pi = \langle \Pi^C_\iota, R_\iota\rangle_{\iota \in I}$ satisfy conditions (LP) and (MAYP). If $c \in \Pi^C$, then $\langle \mathcal{R}_\infty, \mathcal{R}_{co}\rangle \vdash_\nu c \Rightarrow r_\infty$, for some $r_\infty \in R_\infty$.

Proof: First note that, thanks to Theorem 5.27, the statement is equivalent to the following:

> If $c \in \Pi^C$ and $\mathcal{R} \nvdash_\mu c \Rightarrow$, then $\langle \mathcal{R}_\infty, \mathcal{R}_{co}\rangle \vdash_\nu c \Rightarrow \infty$.

Then, the thesis follows by Lemma 5.28. We set $\mathcal{S} = \{c \in C \mid c \in \Pi^C$ and $\mathcal{R} \nvdash_\mu c \Rightarrow\}$, and show that, for all $c \in \mathcal{S}$, there are $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r)$ and $k \in 1..n$ such that, for all $h < k$, $\mathcal{R} \vdash_\mu j_h$ and $C(j_k) \in \mathcal{S}$.

Consider $c \in \mathcal{S}$, then, by (MAYP) (cf. Definition 5.39), there is $\rho = \mathsf{rule}(j_1 \ldots j_n,\ c,\ r)$. By definition of $\mathcal{S}$, we have $\mathcal{R} \nvdash_\mu c \Rightarrow$, hence there exists a (first) $k \in 1..n + 1$ such that $\mathcal{R} \nvdash_\mu j_k$, since, otherwise, we would have $\mathcal{R} \vdash_\mu c \Rightarrow r$. Then, since k is the first index with such property, for all $h < k$, we have $\mathcal{R} \vdash_\mu j_h$, hence, again by condition (MAYP) (cf. Definition 5.39), we have that $\mathcal{R} \nvdash_\mu C(j_k) \Rightarrow$. Finally, since $c \in \Pi^C$ and, for all $h < k$, we have $\mathcal{R} \vdash_\mu j_h$, by Proposition 5.35 we get $C(j_k) \in \Pi^C$, hence $C(j_k) \in \mathcal{S}$, as needed. □

Note that conditions (LP) and (MAYP) actually ensure strong soundness, because, by Lemma 5.34, which is applicable since we assume (LP), we have that converging computations preserve indexes of the predicate.

$$T \quad ::= \quad \mathtt{Nat} \mid T_1 \to T_2 \quad \text{types}$$

$$(\textsc{t-var})\ \frac{}{\Gamma \vdash x : T}\ \Gamma(x) = T \qquad (\textsc{t-const})\ \frac{}{\Gamma \vdash n : \mathtt{Nat}}$$

$$(\textsc{t-abs})\ \frac{\Gamma\{T'/x\} \vdash e : T}{\Gamma \vdash \lambda x.e : T' \to T} \qquad (\textsc{t-app})\ \frac{\Gamma \vdash e_1 : T' \to T \quad \Gamma \vdash e_2 : T'}{\Gamma \vdash e_1\ e_2 : T}$$

$$(\textsc{t-succ})\ \frac{\Gamma \vdash e : \mathtt{Nat}}{\Gamma \vdash \mathtt{succ}\ e : \mathtt{Nat}} \qquad (\textsc{t-choice})\ \frac{\Gamma \vdash e_1 : T \quad \Gamma \vdash e_2 : T}{\Gamma \vdash e_1 \oplus e_2 : T}$$

FIGURE 5.8 λ-calculus: type system

5.6 Examples of soundness proofs

In this section, we show how to use the technique introduced in Section 5.5 to prove soundness of a type system with respect to a big-step semantics, by several examples. We focus on the technique for soundness-must, as it is the usual notion of soundness for type systems. Section 5.6.1 explains in detail how a typical soundness proof can be rephrased in terms of our technique, by reasoning directly on big-step rules. Section 5.6.2 shows a case where this is advantageous, since the property to be checked is *not preserved* by intermediate computation steps, whereas it holds for the whole computation. Section 5.6.3 considers a more sophisticated type system, with intersection and union types. Section 5.6.4 shows another example where types are not preserved, whereas soundness can be proved with our technique. This example is intended as a preliminary step towards a more challenging case. Finally, Section 5.6.5 shows that our technique can also deal with imperative features.

5.6.1 Simply-typed λ-calculus with recursive types

As a first example, we take the λ-calculus with natural constants, successor, and non-deterministic choice introduced in Figure 5.1. We consider a standard simply-typed version with recursive types, obtained by interpreting the production in Figure 5.8 coinductively. Introducing recursive types makes the calculus non-normalising and permits to write interesting programs such as Ω (see Section 5.3.1).

The typing rules are recalled in Figure 5.8. Type environments, written Γ, are finite maps from variables to types, and $\Gamma\{T/x\}$ denotes the map which returns T on x and coincides with Γ elsewhere. We write $\vdash e : T$ for $\emptyset \vdash e : T$.

Let $\langle C_1, R_1, \mathcal{R}_1 \rangle$ be the big-step semantics described in Figure 5.1 (C_1 is the set of expressions and R_1 is the set of values), and let $\Pi 1^C_T = \{e \in C_1 \mid \vdash e : T\}$ and $\Pi 1^R_T = \{v \in R_1 \mid \vdash v : T\}$, where T is a type, defined in Figure 5.8, that is, $\Pi 1^C_T$ and $\Pi 1^R_T$ are the sets of configurations and values of type T, respectively. To prove the three conditions (LP), ($\exists$P) and ($\forall$P) of Section 5.5.2, we need lemmas of inversion, substitution and canonical forms, as in the standard

technique for small-step semantics.

LEMMA 5.42 (Inversion): The following hold:

1. If $\Gamma \vdash x : T$, then $\Gamma(x) = T$.
2. If $\Gamma \vdash n : T$, then $T = \texttt{Nat}$.
3. If $\Gamma \vdash \lambda x.e : T$, then $T = T_1 \to T_2$ and $\Gamma\{T_1/x\} \vdash e : T_2$.
4. If $\Gamma \vdash e_1\ e_2 : T$, then $\Gamma \vdash e_1 : T' \to T$, and $\Gamma \vdash e_2 : T'$.
5. If $\Gamma \vdash \texttt{succ}\ e : T$, then $T = \texttt{Nat}$ and $\Gamma \vdash e : \texttt{Nat}$.
6. If $\Gamma \vdash e_1 \oplus e_2 : T$, then $\Gamma \vdash e_i : T$ with $i \in 1, 2$.

LEMMA 5.43 (Substitution): If $\Gamma\{T'/x\} \vdash e : T$ and $\Gamma \vdash e' : T'$, then $\Gamma \vdash e[e'/x] : T$.

LEMMA 5.44 (Canonical Forms): The following hold:

1. If $\vdash v : T' \to T$, then $v = \lambda x.e$.
2. If $\vdash v : \texttt{Nat}$, then $v = n$.

THEOREM 5.45 (Soundness): The big-step semantics $\langle C_1, R_1, \mathcal{R}_1 \rangle$ and the indexed predicate $\Pi 1$ satisfy the conditions (LP), ($\exists$P) and ($\forall$P) of Section 5.5.2.

Proof: Since the aim of this first example is to illustrate the proof technique, we provide a proof where we explain the reasoning in detail.

PROOF OF (LP): We should prove this condition for each (instantiation of meta-)rule in Figure 5.1.

Case: (APP) Assume that $\vdash e_1\ e_2 : T$ holds. We have to find types for the premises. We proceed as follows:

1. First premise: by Lemma 5.42 (4), $\vdash e_1 : T' \to T$.
2. Second premise: again by Lemma 5.42 (4), $\vdash e_2 : T'$ (without needing the assumption $\vdash \lambda x.e : T' \to T$).
3. Third premise: $\vdash e[v_2/x] : T$ should hold (assuming $\vdash \lambda x.e : T' \to T$, $\vdash v_2 : T'$). Since $\vdash \lambda x.e : T' \to T$, by Lemma 5.42 (3) we have $x{:}T' \vdash e : T$, so by Lemma 5.43 and $\vdash v_2 : T'$ we have $\vdash e[v_2/x] : T$.

Finally, we have to show $\vdash v : T$, assuming $\vdash \lambda x.e : T' \to T$, $\vdash v_2 : T'$ and $\vdash v : T$, which is trivial from the third assumption.

Case: (SUCC) Assume that $\vdash \texttt{succ}\ e : T$ holds. By Lemma 5.42 (5), $T = \texttt{Nat}$, and $\vdash e : \texttt{Nat}$, hence we find Nat as type for the premise. Moreover, $\vdash n + 1 : \texttt{Nat}$ holds by rule (T-CONST).

Case: (CHOICE) Assume that $\vdash e_1 \oplus e_2 : T$ holds. By Lemma 5.42 (6), we have $\vdash e_i : T$, with $i \in 1, 2$. Hence we find T as type for the premise. Finally,

we have to show $\vdash v : T$, assuming $\vdash v : T$, which is trivial.

Case: *(VAL)* Trivial by assumption.

PROOF OF (∃P): We should prove that, for each configuration (here, expression e) such that $\vdash e : T$ holds for some T, there is a rule with this configuration in the conclusion. The expression e cannot be a variable, since a variable cannot be typed in the empty environment. Application, successor, choice, abstraction and constants appear as consequence in the big-step rules (APP), (SUCC), (CHOICE) and (VAL).

PROOF OF (∀P): We should prove this condition for each (instantiation of meta-)rule.

Case: *(APP)* Assuming $\vdash e_1\ e_2 : T$, again by Lemma 5.42 (4) we get $\vdash e_1 : T' \to T$.

1. First premise: if $e_1 \Rightarrow v$ is derivable, then there should be a rule with $e_1\ e_2$ in the conclusion and $e_1 \Rightarrow v$ as first premise. Since we proved (LP), by preservation (Lemma 5.34) $\vdash v : T' \to T$ holds. Then, by Lemma 5.44 (1), v has shape $\lambda x.e$, hence the required rule exists. As noted at page 99, in practice checking (∀P) for a (meta-)rule amounts to show that configurations in the premises evaluate to results which have the required shape (to be a λ-abstraction in this case).

2. Second premise: if $e_1 \Rightarrow \lambda x.e$, and $e_2 \Rightarrow v$, then there should be a rule with $e_1\ e_2$ in the conclusion and $e_1 \Rightarrow \lambda x.e$, $e_2 \Rightarrow v$ as first two premises. This is trivial since the meta-variable v_2 can be freely instantiated in the meta-rule.

3. Third premise: trivial as the previous one.

Case: *(SUCC)* Assuming $\vdash \texttt{succ}\, e : T$, again by Lemma 5.42 (5) we get $\vdash e : \texttt{Nat}$. If $e \Rightarrow v$ is derivable, there should be a rule with $\texttt{succ}\, e$ in the conclusion and $e \Rightarrow v$ as first premise. Indeed, by preservation (Lemma 5.34) and Lemma 5.44 (2), v has shape n.

Case: *(CHOICE)* Trivial since the meta-variable v can be freely instantiated.

Case: *(VAL)* Empty, because there are no premises.

□

An interesting remark is that, differently from the standard approach, there is *no induction* in the proof: everything is *by cases*. This is a consequence of the fact that, as discussed in Section 5.5.2, the three conditions are *local*, that is, they are conditions on single rules. Induction is "hidden" once and for all in the proof that those three conditions are sufficient to ensure soundness.

If we drop in Figure 5.1 rule (SUCC), then condition (∃P) fails, since there is no longer a rule for the well-typed configuration $\texttt{succ}\ n$. If we add the (FOOL) rule $\vdash 0\ 0 : \texttt{Nat}$, then condition (∀P) fails for rule (APP), since $0 \Rightarrow 0$ is derivable, but there is no rule with $0\ 0$ in the conclusion and $0 \Rightarrow 0$ as first premise.

$$
\begin{array}{rcll}
e & ::= & x \mid e.\mathsf{f} \mid \mathsf{new}\ \mathsf{C}(e_1, \ldots, e_n) \mid e.\mathsf{m}(e_1, \ldots, e_n) \mid \lambda xs.e \mid (T)\,e & \text{expression} \\
T & ::= & \mathsf{C} \mid \mathsf{I} & \text{type}
\end{array}
$$

$$
\begin{array}{rcll}
c & ::= & \langle \mathrm{E}, e \rangle & \text{configuration} \\
v & ::= & [vs]^{\mathsf{C}} \mid \lambda xs.e & \text{result (value)}
\end{array}
$$

$$
(\textsc{var})\ \frac{}{\langle \mathrm{E}, x \rangle \Rightarrow v}\ \ \mathrm{E}(x) = v \qquad (\textsc{new})\ \frac{\langle \mathrm{E}, e_i \rangle \Rightarrow v_i \quad \forall i \in 1..n}{\langle \mathrm{E}, \mathsf{new}\ \mathsf{C}(e_1, \ldots, e_n) \rangle \Rightarrow [v_1, \ldots, v_n]^{\mathsf{C}}}
$$

$$
(\textsc{field-access})\ \frac{\langle \mathrm{E}, e \rangle \Rightarrow [v_1, \ldots, v_n]^{\mathsf{C}}}{\langle \mathrm{E}, e.\mathsf{f}_i \rangle \Rightarrow v_i}\ \ \begin{array}{l} \mathsf{fields}(\mathsf{C}) = T_1\,\mathsf{f}_1\,; \ldots T_n\,\mathsf{f}_n\,; \\ i \in 1..n \end{array}
$$

$$
(\textsc{invk})\ \frac{\begin{array}{l} \langle \mathrm{E}, e_0 \rangle \Rightarrow [vs]^{\mathsf{C}} \\ \langle \mathrm{E}, e_i \rangle \Rightarrow v_i \quad \forall i \in 1..n \\ \langle x_1{:}v_1, \ldots, x_n{:}v_n, \mathtt{this}{:}[vs]^{\mathsf{C}}, e \rangle \Rightarrow v \end{array}}{\langle \mathrm{E}, e_0.\mathsf{m}(e_1, \ldots, e_n) \rangle \Rightarrow v}\ \ \mathsf{mbody}(\mathsf{C}, \mathsf{m}) = \langle x_1 \ldots x_n, e \rangle
$$

$$
(\lambda\textsc{-invk})\ \frac{\begin{array}{l} \langle \mathrm{E}, e_0 \rangle \Rightarrow \lambda xs.e \\ \langle \mathrm{E}, e_i \rangle \Rightarrow v_i \quad \forall i \in 1..n \\ \langle x_1{:}v_1, \ldots, x_n{:}v_n, e \rangle \Rightarrow v \end{array}}{\langle \mathrm{E}, e_0.\mathsf{m}(e_1, \ldots, e_n) \rangle \Rightarrow v}
$$

$$
(\lambda)\ \frac{}{\langle \mathrm{E}, \lambda xs.e \rangle \Rightarrow \lambda xs.e} \qquad (\textsc{upcast})\ \frac{\langle \mathrm{E}, e \rangle \Rightarrow v}{\langle \mathrm{E}, (T)\,e \rangle \Rightarrow v}
$$

FIGURE 5.9 MINIFJ&λ: syntax and big-step semantics

5.6.2 MINIFJ&λ

In this example, the language is a subset of FJ&λ (Bettini et al., 2018), a calculus extending Featherweight Java (FJ) with λ-abstractions and intersection types, introduced in Java 8. To keep the example small, we do not consider intersections and focus on one key typing feature: λ-abstractions can only be typed when occurring in a context requiring a given type (called the *target type*). In a small-step semantics, this poses a problem: reduction can move λ-abstractions into arbitrary contexts, leading to intermediate terms which would be ill-typed. To maintain subject reduction, Bettini et al. (2018) decorate λ-abstractions with their initial target type. In a big-step semantics, there is no need of intermediate terms and annotations.

The syntax is given in the first part of Figure 5.9. We assume sets of *variables* x, *class names* C, *interface names* I, J, *field names* f, and *method names* m. Interfaces which have *exactly* one method (dubbed *functional interfaces*) can be used as target types. Expressions are those of FJ, plus λ-abstractions, and types are class and interface names. In $\lambda xs.e$ we assume that xs is not empty and e is not a λ-abstraction. For simplicity, we only consider *upcasts*, which have no runtime effect, but are important to allow the programmer to use λ-abstractions, as exemplified in discussing typing rules.

To be concise, the class table is abstractly modelled as follows:

- fields(C) gives the sequence of field declarations $T_1\,\mathsf{f}_1\,;..T_n\,\mathsf{f}_n\,;$ for class C
- mtype(T, m) gives, for each method m in class or interface T, the pair $T_1 \ldots T_n \to T'$ consisting of the parameter types and return type
- mbody(C, m) gives, for each method m in class C, the pair $\langle x_1 \ldots x_n, e\rangle$ consisting of the parameters and body
- <: is the reflexive and transitive closure of the union of the extends and implements relations
- !mtype(I) gives, for each *functional* interface I, mtype(I, m), where m is the only method of I.

The big-step semantics is given in the last part of Figure 5.9. MiniFJ&λ shows an example of instantiation of the framework where configurations include an auxiliary structure, rather than being just language terms. In this case, the structure is an *environment* E (a finite map from variables to values) modelling the current stack frame. Furthermore, results are not particular configurations: they are either *objects*, of shape $[vs]^{\mathsf{C}}$, or λ-abstractions.

Throughout this section *xs* and *vs* denote lists of variables and values, respectively. Rules for FJ constructs are straightforward. Note that, since we only consider upcasts, casts have no runtime effect. Indeed, they are guaranteed to succeed on well-typed expressions. Rule (λ-INVK) shows that, when the receiver of a method is a λ-abstraction, the method name is not significant at runtime, and the effect is that the body of the function is evaluated as in the usual application.

The type system is given in Figure 5.10. The following assumptions formalize standard FJ typing constraints on the class table.

(FJ1) Method bodies are well-typed with respect to method types:
- either mbody(C, m) and mtype(C, m) are both undefined
- or mbody(C, m) = $\langle x_1 \ldots x_n, e\rangle$, mtype(C, m) = $T_1 \ldots T_n \to T$, and $x_1{:}T_1, \ldots, x_n{:}T_n$, this:C $\vdash e : T$.

(FJ2) Fields are inherited, no field hiding:
if $T <: T'$, and fields(T') = $T_1\,\mathsf{f}_1\,; \ldots T_n\,\mathsf{f}_n\,;$, then fields($T$) = $T_1\,\mathsf{f}_1\,; \ldots T_m\,\mathsf{f}_m\,;$, $m \geq n$, and $\mathsf{f}_i \neq \mathsf{f}_j$ for $i \neq j$.

(FJ3) Methods are inherited, no method overloading, invariant overriding:
if $T <: T'$, and mtype(T', m) is defined, then mtype(T, m) = mtype(T', m).

Besides the standard typing features of FJ, the MiniFJ&λ type system ensures the following.

- A functional interface I can be assigned as type to a λ-abstraction which has the functional type of the method, see rule (τ-λ).
- A λ-abstraction should have a *target type* determined by the context where the λ-abstraction occurs. More precisely, as described by Gosling et al. (2014, p. 602), a λ-abstraction in our calculus can only occur as return expression of a method or argument of constructor, method call

$$\text{(T-CONF)}\ \frac{\vdash v_i : T_i \quad \forall i \in 1..n \qquad x_1{:}T'_1, \ldots, x_n{:}T'_n \vdash e : T}{\vdash \langle x_1{:}v_1, \ldots, x_n{:}v_n, e\rangle : T} \quad T_i <: T'_i \quad \forall i \in 1..n$$

$$\text{(T-VAR)}\ \frac{}{\Gamma \vdash x : T} \quad \Gamma(x) = T \qquad \text{(T-UPCAST)}\ \frac{\Gamma \vdash e : T}{\Gamma \vdash (T)\,e : T}$$

$$\text{(T-FIELD-ACCESS)}\ \frac{\Gamma \vdash e : \mathsf{C}}{\Gamma \vdash e.\mathsf{f} : T_i} \quad \begin{array}{l}\mathsf{fields}(\mathsf{C}) = T_1\,\mathsf{f}_1\,; \ldots T_n\,\mathsf{f}_n\,; \\ i \in 1..n\end{array}$$

$$\text{(T-NEW)}\ \frac{\Gamma \vdash e_i : T_i \quad \forall i \in 1..n}{\Gamma \vdash \mathsf{new}\,\mathsf{C}(e_1, \ldots, e_n) : \mathsf{C}} \quad \mathsf{fields}(\mathsf{C}) = T_1\,\mathsf{f}_1\,; \ldots T_n\,\mathsf{f}_n\,;$$

$$\text{(T-INVK)}\ \frac{\Gamma \vdash e_i : T_i \quad \forall i \in 0..n}{\Gamma \vdash e_0.\mathsf{m}(e_1, \ldots, e_n) : T} \quad \begin{array}{l}e_0 \text{ not of shape } \lambda xs.e \\ \mathsf{mtype}(T_0, \mathsf{m}) = T_1 \ldots T_n \to T\end{array}$$

$$\text{(T-}\lambda\text{)}\ \frac{x_1{:}T_1, \ldots, x_n{:}T_n \vdash e : T}{\Gamma \vdash \lambda xs.e : \mathsf{I}} \quad !\mathsf{mtype}(\mathsf{I}) = T_1 \ldots T_n \to T$$

$$\text{(T-OBJECT)}\ \frac{\Gamma \vdash v_i : T'_i \quad \forall i \in 1..n}{\Gamma \vdash [v_1, \ldots, v_n]^{\mathsf{C}} : \mathsf{C}} \quad \begin{array}{l}\mathsf{fields}(\mathsf{C}) = T_1\,\mathsf{f}_1\,; \ldots T_n\,\mathsf{f}_n\,; \\ T'_i <: T_i \quad \forall i \in 1..n\end{array}$$

$$\text{(T-SUB)}\ \frac{\Gamma \vdash e : T}{\Gamma \vdash e : T'} \quad \begin{array}{l}e \text{ not of shape } \lambda xs.e \\ T <: T'\end{array}$$

FIGURE 5.10 MINIFJ&λ: type system

or cast. Then, in some contexts a λ-abstraction cannot be typed, in our calculus when occurring as receiver in field access or method invocation, hence these cases should be prevented. This is implicit in rule (T-FIELD-ACCESS), since the type of the receiver should be a class name, whereas it is explicitly forbidden in rule (T-INVK). For the same reason, a λ-abstraction cannot be the main expression to be evaluated.

- A λ-abstraction with a given target type J should have type *exactly* J: a subtype I of J is not enough. Consider, for instance, the following program:

```
interface J {}
interface I extends J { A m(A x); }
class C {
  C m(I y) { return new C().n(y); }
  C n(J y) { return new C(); }
}
```

and the main expression `new C().n(`$\lambda x.x$`)`. Here, the λ-abstraction has target type J, which is *not* a functional interface, hence the expression is ill-typed in Java (the compiler has no functional type against which to typecheck the λ-abstraction). On the other hand, in the body of method m, the parameter y of type I can be passed, as usual, to method n expecting a supertype. For instance, the main expression `new C().m(`$\lambda x.x$`)` is well-typed, since the λ-abstraction has target type I, and can be safely passed to method n, since it is not used as function there. To formalise this

behaviour, it is forbidden to apply subsumption to λ-abstractions, see rule (T-SUB).

- However, λ-abstractions occurring as results rather than in source code (that is, in the environment and as fields of objects) are allowed to have a subtype of the required type, see the explicit side condition in rules (T-CONF) and (T-OBJECT). For instance, if C is a class with one field J f, the expression $\mathsf{new\ C((I)}\lambda x.x\mathsf{)}$ is well-typed, whereas $\mathsf{new\ C(}\lambda x.x\mathsf{)}$ is ill typed, since rule (T-SUB) cannot be applied to λ-abstractions. When the expression is evaluated, the result is $[\lambda x.x]^{\mathsf{C}}$, which is well-typed.

As mentioned at the beginning, the obvious small-step semantics would produce not typable expressions. In the above example, we get

$$\mathsf{new\ C((I)}\lambda x.x\mathsf{)} \longrightarrow \mathsf{new\ C(}\lambda x.x\mathsf{)} \longrightarrow [\lambda x.x]^{\mathsf{C}}$$

and $\mathsf{new\ C(}\lambda x.x\mathsf{)}$ has no type, while $\mathsf{new\ C((I)}\lambda x.x\mathsf{)}$ and $[\lambda x.x]^{\mathsf{C}}$ have type C.

As expected, to show soundness (Theorem 5.48) lemmas of inversion and canonical forms are handy: they can be easily proved as usual. Instead, we do not need a substitution lemma, since environments associate variables with values.

LEMMA 5.46 (Inversion): The following hold:

1. If $\Gamma \vdash \langle x_1{:}v_1, \ldots, x_n{:}v_n, e\rangle : T$, then $\vdash v_i :<: T_i$ for all $i \in 1..n$ and $x_1{:}T_1, \ldots, x_n{:}T_n \vdash e : T$.
2. If $\Gamma \vdash x : T$, then $\Gamma(x) <: T$.
3. If $\Gamma \vdash e.\mathsf{f}_i : T$, then $\Gamma \vdash e : \mathsf{C}$ and $\mathsf{fields(C)} = T_1\,\mathsf{f}_1\mathsf{;}\ \ldots\ T_n\,\mathsf{f}_n\mathsf{;}$ and $T_i <: T$ where $i \in 1..n$.
4. If $\Gamma \vdash \mathsf{new\ C}(e_1, \ldots, e_n) : T$, then $\mathsf{C} <: T$ and $\mathsf{fields(C)} = T_1\,\mathsf{f}_1\mathsf{;}\ \ldots\ T_n\,\mathsf{f}_n\mathsf{;}$ and $\Gamma \vdash e_i : T_i$ for all $i \in 1..n$.
5. If $\Gamma \vdash e_0.\mathsf{m}(e_1, \ldots, e_n) : T$, then e_0 not of shape $\lambda xs.e$ and $\Gamma \vdash e_i : T_i$ for all $i \in 0..n$ and $\mathsf{mtype}(T_0, \mathsf{m}) = T_1 \ldots T_n \to T'$ with $T' <: T$.
6. If $\Gamma \vdash \lambda xs.e : T$, then $T = \mathsf{I}$ and $\mathsf{!mtype(I)} = T_1 \ldots T_n \to T'$ and $x_1{:}T_1, \ldots, x_n{:}T_n \vdash e : T'$.
7. If $\Gamma \vdash (T')\,e : T$, then $\Gamma \vdash e : T'$ and $T' <: T$.
8. If $\Gamma \vdash [v_1, \ldots, v_n]^{\mathsf{C}} : T$, then $\mathsf{C} <: T$ and $\mathsf{fields(C)} = T_1\,\mathsf{f}_1\mathsf{;}\ \ldots\ T_n\,\mathsf{f}_n\mathsf{;}$ and $\Gamma \vdash v_i : T'_i$ and $T'_i <: T_i$ for all $i \in 1..n$.

LEMMA 5.47 (Canonical Forms): The following hold:

1. If $\vdash v : \mathsf{C}$, then $v = [vs]^{\mathsf{D}}$ and $\mathsf{D} <: \mathsf{C}$.
2. If $\vdash v : \mathsf{I}$, then either $v = [vs]^{\mathsf{C}}$ and $\mathsf{C} <: \mathsf{I}$ or $v = \lambda xs.e$ and I is a functional interface.

We write $\Gamma \vdash e :<: T$ as short for $\Gamma \vdash e : T'$ and $T' <: T$ for some T'. In order to state soundness, set $\langle C_2, R_2, \mathcal{R}_2 \rangle$ the big-step semantics defined in Figure 5.9, and let $\Pi 2^C_T = \{\langle \mathrm{E}, e\rangle \in C_2 \mid\vdash \langle \mathrm{E}, e\rangle :<: T\}$ and $\Pi 2^R_T = \{v \in R_2 \mid\vdash v :<: T\}$, for T defined in Figure 5.9.

THEOREM 5.48 (Soundness): The big-step semantics $\langle C_2, R_2, \mathcal{R}_2 \rangle$ and the indexed predicate $\Pi 2$ satisfy the conditions (LP), ($\exists$P) and ($\forall$P) of Section 5.5.2.

Proof: PROOF OF (LP): The proof is by cases on instantiations of meta-rules. Considering a rule with typed conclusion $\langle y_1{:}\hat{v}_1, \ldots, y_p{:}\hat{v}_p, e\rangle$, Lemma 5.46 (1) implies $\vdash \hat{v}_\ell :<: \hat{T}_\ell$ for all $\ell \in 1..p$ and $y_1{:}\hat{T}_1, \ldots, y_p{:}\hat{T}_p \vdash e :<: T$ for some $\hat{T}_1, \ldots, \hat{T}_p$.

Case: (VAR) Lemma 5.46 (1) gives $\vdash \mathrm{E}(x) :<: T'$ and $x{:}T' \vdash x : T$. Lemma 5.46 (2) implies $T' <: T$, so we conclude $\vdash \mathrm{E}(x) :<: T$ by transitivity of $<:$.

Case: (FIELD-ACCESS) Lemma 5.46 (3) applied to $\Gamma \vdash e.\mathsf{f}_i : T$ implies $\Gamma \vdash e : \mathtt{D}$ and $\mathsf{fields}(\mathtt{D}) = T_1\,\mathsf{f}_1\,; \ldots T_m\,\mathsf{f}_m\,;$ and $T_i <: T$ where $i \in 1..m$. Since $\langle \mathrm{E}, e\rangle \Rightarrow [v_1, \ldots, v_n]^{\mathtt{C}}$ is a premise we assume $\vdash [v_1, \ldots, v_n]^{\mathtt{C}} :<: \mathtt{D}$, which implies $\mathtt{C} <: \mathtt{D}$ and $\mathsf{fields}(\mathtt{C}) = T'_1\,\mathsf{f}'_1\,; \ldots T'_n\,\mathsf{f}'_n\,;$ and $\Gamma \vdash v_j :<: T'_j$ for all $j \in 1..n$ by Lemma 5.46 (8). From $\mathtt{C} <: \mathtt{D}$ and assumption (FJ2) we have $m \leq n$ and $T_j = T'_j$ and $\mathsf{f}_j = \mathsf{f}'_j$ for all $j \in 1..m$. We conclude $\vdash v_i :<: T$.

Case: (NEW) Lemma 5.46 (4) applied to $\Gamma \vdash \mathtt{new}\ \mathtt{C}(e_1, \ldots, e_n) : T$ implies $\mathtt{C} <: T$ and $\mathsf{fields}(\mathtt{C}) = T_1\,\mathsf{f}_1\,; \ldots T_n\,\mathsf{f}_n\,;$ and $\Gamma \vdash e_i : T_i$ for all $i \in 1..n$. Since $\langle \mathrm{E}, e_i\rangle \Rightarrow v_i$ is a premise we assume $\vdash v_i :<: T_i$ for all $i \in 1..n$. Using rule (T-OBJECT) we derive $\vdash [v_1, \ldots, v_n]^{\mathtt{C}} :<: T$.

Case: (INVK) Lemma 5.46 (5) applied to $\Gamma \vdash e_0.\mathsf{m}(e_1, \ldots, e_n) : T$ implies e_0 not of shape $\lambda xs.e$ and $\Gamma \vdash e_i : T_i$ for all $i \in 0..n$ and $\mathsf{mtype}(T_0, \mathsf{m}) = T_1 \ldots T_n \to T'$ with $T' <: T$. Since $\langle \mathrm{E}, e_0\rangle \Rightarrow [vs']^{\mathtt{C}}$ is a premise we assume $\vdash [vs']^{\mathtt{C}} :<: T_0$, which implies $\mathtt{C} <: T_0$ by Lemma 5.46 (8). Since $\langle \mathrm{E}, e_i\rangle \Rightarrow v_i$ is a premise we assume $\vdash v_i :<: T_i$ for all $i \in 1..n$. We have $\mathsf{mtype}(\mathtt{C}, \mathsf{m}) = T_1 \ldots T_n \to T'$ since $\mathsf{mtype}(T_0, \mathsf{m}) = T_1 \ldots T_n \to T'$ and $\mathtt{C} <: T_0$ by assumption (FJ3). By assumption (FJ1), $x_1{:}T_1, \ldots, x_n{:}T_n, \mathtt{this}{:}\mathtt{C} \vdash e : T'$. Therefore, by rule (T-CONF) and since $T' <: T$, we can derive $\vdash \langle x_1{:}v_1, \ldots, x_n{:}v_n, \mathtt{this}{:}[vs']^{\mathtt{C}}, e\rangle :<: T$.

Case: (λ-INVK) Lemma 5.46 (5) applied to $\Gamma \vdash e_0.\mathsf{m}(e_1, \ldots, e_n) : T$ implies e_0 not of shape $\lambda xs.e'$ and $\Gamma \vdash e_i : T_i$ for all $i \in 0..n$ and $\mathsf{mtype}(T_0, \mathsf{m}) = T_1 \ldots T_n \to T'$ with $T' <: T$. Since $\langle \mathrm{E}, e_0\rangle \Rightarrow \lambda xs.e$ is a premise we assume $\vdash \lambda xs.e :<: T_0$, which implies $\mathsf{I} <: T_0$ and $!\mathsf{mtype}(\mathsf{I}) = T_1 \ldots T_n \to T'$ and $x_1{:}T_1, \ldots, x_n{:}T_n \vdash e : T'$ by Lemma 5.46 (6). Since $\langle \mathrm{E}, e_i\rangle \Rightarrow v_i$ is a premise we assume $\vdash v_i :<: T_i$ for all $i \in 1..n$. Therefore we derive $\vdash \langle x_1{:}v_1, \ldots, x_n{:}v_n, e\rangle :<: T$.

Case: (UPCAST) Lemma 5.46 (7) applied to $\Gamma \vdash (T')\,e : T$ implies $\Gamma \vdash e :<: T$. From $\langle \mathrm{E}, e\rangle \Rightarrow v$ we conclude $\vdash v :<: T$.

$$T \quad ::= \quad \mathtt{Nat} \mid T_1 \to T_2 \mid T_1 \wedge T_2 \mid T_1 \vee T_2 \quad \text{type}$$

$$(\wedge\,\mathrm{I})\ \frac{\Gamma \vdash e : T \quad \Gamma \vdash e : S}{\Gamma \vdash e : T \wedge S} \qquad (\wedge\,\mathrm{E})\ \frac{\Gamma \vdash e : T \wedge S}{\Gamma \vdash e : T} \qquad (\wedge\,\mathrm{E})\ \frac{\Gamma \vdash e : T \wedge S}{\Gamma \vdash e : S}$$

$$(\vee\,\mathrm{I})\ \frac{\Gamma \vdash e : T}{\Gamma \vdash e : T \vee S} \qquad (\vee\,\mathrm{I})\ \frac{\Gamma \vdash e : S}{\Gamma \vdash e : T \vee S}$$

FIGURE 5.11 Intersection and union types: syntax and typing rules

PROOF OF ($\exists$P): It is easy to verify that if $\vdash \langle \mathrm{E}, e\rangle :<: T$, then there is a rule in Figure 5.9, whose conclusion is $\langle \mathrm{E}, e\rangle$, just because for every sytactic construct there is a corresponding rule. The only non-trivial case is that of variables: if $\vdash \langle \mathrm{E}, x\rangle :<: T$, then by Lemma 5.46 (1,2), $x \in \mathrm{dom}\mathrm{E}$, hence rule (VAR) is applicable, as the side condition is satisfied.

PROOF OF ($\forall$P): Rule (FIELD-ACCESS) requires that $\langle \mathrm{E}, e\rangle$ reduces to an object, and this is assured by the typing rule (T-FIELD-ACCESS), which prescribes a class type for the expression e, together with the validity of condition (LP) (which assures type preservation by Lemma 5.34) and Lemma 5.47 (1). For a well-typed method call $e_0.\mathtt{m}(e_1, \ldots, e_n)$ the configuration $\langle \mathrm{E}, e_0\rangle$ can reduce either to an object or to a λ-expression. In the first case we can apply rule (INVK) and in the second case rule (λ-INVK). In both cases the typing assures that the arguments are in the right number. The condition holds for the last premise of rule (INVK) by the well-typing of the class table. The condition holds for the last premise of rule (λ-INVK) by rule (T-λ). □

5.6.3 Intersection and union types

We enrich the type system of Figure 5.8 by adding intersection and union type constructors and the corresponding typing rules, see Figure 5.11. As usual we require an infinite number of arrows in each infinite path for the trees representing types. Intersection types for the λ-calculus have been widely studied, *e.g.*, by Barendregt, Dekkers, and Statman (2013). Union types naturally model conditionals (Grudzinski, 2000) and non-deterministic choice (Dezani-Ciancaglini, de'Liguoro, and Piperno, 1998).

The typing rules for the introduction and the elimination of intersection and union are standard, except for the absence of the union elimination rule:

$$(\vee E)\ \frac{\Gamma\{T/x\} \vdash e : V \quad \Gamma\{S/x\} \vdash e : V \quad \Gamma \vdash e' : T \vee S}{\Gamma \vdash e[e'/x] : V}$$

As a matter of fact, rule ($\vee E$) is unsound for $\oplus$. For example, let split the type `Nat` into `Even` and `Odd` and add the expected typings for natural numbers. The prefix addition + has type $(\mathtt{Even} \to \mathtt{Even} \to \mathtt{Even}) \wedge (\mathtt{Odd} \to \mathtt{Odd} \to \mathtt{Even})$ and we derive

$$\dfrac{x{:}\mathtt{Even} \vdash +\,x\,x{:}\mathtt{Even} \quad x{:}\mathtt{Odd} \vdash +\,x\,x{:}\mathtt{Even} \quad \dfrac{\dfrac{\vdash 1 : \mathtt{Odd}}{\vdash 1 : \mathtt{Even} \vee \mathtt{Odd}} \quad \dfrac{\vdash 2 : \mathtt{Even}}{\vdash 2 : \mathtt{Even} \vee \mathtt{Odd}}}{\vdash (1 \oplus 2) : \mathtt{Even} \vee \mathtt{Odd}}}{\vdash +(1 \oplus 2)(1 \oplus 2) : \mathtt{Even}}$$

We cannot assign the type `Even` to 3, which is a possible result, so strong soundness is lost. In addition, in the small-step approach, we cannot assign `Even` to the intermediate term + 1 2, so subject reduction fails. In the big-step approach, there is no such intermediate term; however, condition (LP) fails for the big-step rule for +. Indeed, considering the following instantiation of the rule:

$$(+)\ \frac{1 \oplus 2 \Rightarrow 1 \quad 1 \oplus 2 \Rightarrow 2}{+(1 \oplus 2)(1 \oplus 2) \Rightarrow 3}$$

and the type `Even` for the conclusion: we cannot assign this type to the final result as required by (LP) (cf. Definition 5.33 (2)).

Intersection types allow to derive meaningful types also for expressions containing variables applied to themselves, for example we can derive $\vdash \lambda x.x\,x : (T \to S) \wedge T \to S$. With union types all non-deterministic choices between typable expressions can be typed too, since we can derive $\Gamma \vdash e_1 \oplus e_2 : T_1 \vee T_2$ from $\Gamma \vdash e_1 : T_1$ and $\Gamma \vdash e_2 : T_2$.

We now state standard lemmas for the type system, which are handy towards the soundness proof. We first define the *subtyping* relation $T \leq S$ as the smallest preorder such that:

- $S \leq T_1$ and $S \leq T_2$ imply $S \leq T_1 \wedge T_2$;
- $T \wedge S \leq T$ and $T \wedge S \leq S$;
- $T \leq T \vee S$ and $T \leq S \vee T$.

It is easy to verify that $T \leq S$ iff $\Gamma, x{:}T \vdash x : S$ for an arbitrary variable x, using rules (∧I), (∧E) and (∨I).

LEMMA 5.49 (Inversion): The following hold:

1. If $\Gamma \vdash x : T$, then $\Gamma(x) \leq T$.
2. If $\Gamma \vdash n : T$, then $\mathtt{Nat} \leq T$.
3. If $\Gamma \vdash \lambda x.e : T$, then $\Gamma\{S_i/x\} \vdash e : V_i$ for $i \in 1..m$ and $\bigwedge_{i \in 1..m}(S_i \to V_i) \leq T$.
4. If $\Gamma \vdash e_1\, e_2 : T$, then $\Gamma \vdash e_1 : S_i \to V_i$ and $\Gamma \vdash e_2 : S_i$ for $i \in 1..m$ and $\bigwedge_{i \in 1..m} V_i \leq T$.
5. If $\Gamma \vdash \mathtt{succ}\, e : T$, then $\mathtt{Nat} \leq T$ and $\Gamma \vdash e : \mathtt{Nat}$.
6. If $\Gamma \vdash e_1 \oplus e_2 : T$, then $\Gamma \vdash e_i : T$ with $i \in 1, 2$.

LEMMA 5.50 (Substitution): If $\Gamma\{T'/x\} \vdash e : T$ and $\Gamma \vdash e' : T'$, then $\Gamma \vdash e[e'/x] : T$.

LEMMA 5.51 (Canonical Forms): The following hold:

1. If $\vdash v : T' \to T$, then $v = \lambda x.e$.
2. If $\vdash v : \mathtt{Nat}$, then $v = n$.

In order to state soundness, let $\Pi 3^C_T = \{e \in C_1 \mid\vdash e : T\}$ and $\Pi 3^R_T = \{v \in R_1 \mid\vdash v : T\}$, for T defined in Figure 5.11.

THEOREM 5.52 (Soundness): The big-step semantics $\langle C_1, R_1, \mathcal{R}_1\rangle$ and the indexed predicate $\Pi 3$ satisfy the conditions (LP), ($\exists$P) and ($\forall$P) of Section 5.5.2.

Proof sketch We prove conditions only for rule (APP), the other cases are similar (cf. proof of Theorem 5.45).

PROOF OF (LP): The proof is by cases on instantiations of meta-rules. For rule (APP) Lemma 5.49 (4) applied to $\vdash e_1\, e_2 : T$ implies $\vdash e_1 : S_i \to V_i$ and $\vdash e_2 : S_i$ for $i \in 1..m$ and $\bigwedge_{i\in 1..m} V_i \leq T$. Now, from assumptions of (LP), we get $\vdash \lambda x.e : S_i \to V_i$ and $\vdash v_2 : S_i$ for $i \in 1..m$. Lemma 5.49 (3) implies $x : S_i \vdash e : V_i$, so by Lemma 5.50 we have $\vdash e[v_2/x] : V_i$ for $i \in 1..m$. We can derive $\vdash e[v_2/x] : T$ using rules ($\wedge$I), ($\wedge$E) and ($\vee$I).

PROOF OF ($\exists$P): The proof is as in Theorem 5.45.

PROOF OF ($\forall$P): The proof is by cases on instantiations of meta-rules. For rule (APP) Lemma 5.49 (4) applied to $\vdash e_1\, e_2 : T$ implies $\vdash e_1 : S_i \to V_i$ for $i \in 1..m$. If $e_1 \Rightarrow v$ we get $\vdash v : S_i \to V_i$ for $i \in 1..m$ by (LP) and Lemma 5.34. Lemma 5.51 (1) applied to $\vdash v : S_i \to V_i$ implies $v = \lambda x.e$ as needed. □

5.6.4 MiniFJ$^\vee$

A well-known example in which proving soundness with respect to small-step semantics is extremely challenging is the standard type system with intersection and union types (Barbanera, Dezani-Ciancaglini, and de'Liguoro, 1995) w.r.t. the pure λ-calculus with full reduction. Indeed, the standard subject reduction technique fails[5], since, for instance, we can derive the type

$$(T \to T \to V) \wedge (S \to S \to V) \to (U \to T \vee S) \to U \to V$$

for both $\lambda x.\lambda y.\lambda z.x\,((\lambda t.t)\,(y\,z))\,((\lambda t.t)\,(y\,z))$ and $\lambda x.\lambda y.\lambda z.x\,(y\,z)\,(y\,z)$, but the intermediate expressions $\lambda x.\lambda y.\lambda z.x\,((\lambda t.t)\,(y\,z))\,(y\,z)$ and $\lambda x.\lambda y.\lambda z.x\,(y\,z)\,((\lambda t.t)\,(y\,z))$ do not have this type.

As the example shows, the key problem is that rule (VE) can be applied to expression e where the same subexpression e' occurs more than once. In the non-deterministic case, as shown by the example in the previous section, this is unsound, since e' can reduce to different values. In the deterministic case, instead, this is sound, but cannot be proved by subject reduction. Since using big-step semantics there are no intermediate steps to be typed, our approach seems very promising to investigate an alternative proof of soundness. Whereas we leave this challenging problem to future work, here as first step we describe a calculus with a much simpler version of the problematic feature.

5 For this reason, Barbanera, Dezani-Ciancaglini, and de'Liguoro (1995) prove soundness by an ad-hoc technique, that is, by considering parallel reduction and an equivalent type system à la Gentzen, which enjoys the cut elimination property.

The calculus is a variant of FJ$\vee$, introduced by Igarashi and Nagira (2007), an extension of FJ (Igarashi, Pierce, and Wadler, 2001) with union types. As discussed more extensively by Igarashi and Nagira (2007), this gives the ability to define a supertype even after a class hierarchy is fixed, grouping independently developed classes with similar interfaces. In fact, given some types, their union type can be viewed as an interface type that "factors out" their common features. With respect to FJ$\vee$, we do not consider cast and type-case constructs and, more importantly, in the typing rules we handle differently union types, taking inspiration directly from rule (vE) of the λ-calculus. With this approach, we enhance the expressivity of the type system, since it becomes possible to eliminate unions simultaneously for an arbitrary number of arguments, including the receiver, in a method invocation, provided that they are all equal to each other. We dub this calculus MiniFJ$^{\vee}$.

Figure 5.12 gives the syntax, big-step semantics and typing rules of MiniFJ$^{\vee}$. The subtyping relation <: is the reflexive and transitive closure of the union of the extends relation and the standard rules for union:

$$\frac{}{T_1 <: T_1 \vee T_2} \qquad \frac{}{T_2 <: T_1 \vee T_2} \qquad \frac{T_1 <: T \quad T_2 <: T}{T_1 \vee T_2 <: T}$$

The functions mtype, fields and mbody are defined as for MiniFJ&λ, apart that here fields, method parameters and return types can be union types as well, still assuming the conditions on the class table (FJ1), (FJ2), and (FJ3).

Clearly rule (T-V-ELIM) is inspired by rule (vE), but restricted only to some specific contexts, named *(union) elimination contexts*. Elimination contexts are field access and method invocation, where the latter has $n > 0$ holes corresponding to the receiver and (for simplicity the first) $n - 1$ parameters. Thanks to this restriction, we are able to prove a standard inversion lemma, which is not known for the general rule in the λ-calculus.

Given an elimination context E, we denote by $E[e]$ the expression obtained by filling all holes of E by e.

This rule allows us to make the type system more "structural", with respect to FJ, similarly to what happens in FJ$\vee$. Let us consider the following classes:

```
class C {
  A f; Object g;
  C update(A x) {...}
  Bool eq(C x) {..}
}
class D {
  A f;
  D update(A x) {...}
  Bool eq(D x) {...}
}
```

They share a common structure, but they are not related by inheritance (there is no common superclass abstracting shared features), hence in standard FJ they cannot be handled uniformly. By means of (T-V-ELIM) this is possible: for instance, we can write a wrapper class that, in a sense, provides the common interface of `C` and `D` "ex-post"

e	::=	$x \mid e.\mathsf{f} \mid \mathsf{new}\ \mathsf{C}(e_1, \ldots, e_n) \mid e.\mathsf{m}(e_1, \ldots, e_n)$ $\mathtt{if}\ e\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2 \mid \mathtt{true} \mid \mathtt{false}$	expression
v	::=	$\mathsf{new}\ \mathsf{C}(v_1, \ldots, v_n) \mid \mathtt{true} \mid \mathtt{false}$	value
T	::=	$\mathsf{C} \mid \mathtt{Bool} \mid T_1 1 \vee T_2$	type
E	::=	$[].\mathsf{f} \mid [].\mathsf{m}([], \ldots, [], e_1, \ldots, e_n)$	elimination context

$$(\textsc{field})\ \frac{e \Rightarrow \mathsf{new}\ \mathsf{C}(v_1, \ldots, v_n)}{e.\mathsf{f}_i \Rightarrow v_i} \quad \begin{array}{l}\mathsf{fields}(\mathsf{C}) = T_1\,\mathsf{f}_1\,; \ldots T_n\,\mathsf{f}_n\,; \\ i \in 1..n\end{array}$$

$$(\textsc{new})\ \frac{e_i \Rightarrow v_i \quad \forall i \in 1..n}{\mathsf{new}\ \mathsf{C}(e_1, \ldots, e_n) \Rightarrow \mathsf{new}\ \mathsf{C}(v_1, \ldots, v_n)}$$

$$(\textsc{invk})\ \frac{\begin{array}{l}e_0 \Rightarrow \mathsf{new}\ \mathsf{C}(vs') \\ e_i \Rightarrow v_i \quad \forall i \in 1..n \\ e[v_1/x_1] \ldots [v_n/x_n][\mathsf{new}\ \mathsf{C}(vs')/\mathtt{this}] \Rightarrow v\end{array}}{e_0.\mathsf{m}(e_1, \ldots, e_n) \Rightarrow v} \quad \mathsf{mbody}(\mathsf{C}, \mathsf{m}) = \langle x_1 \ldots x_n, e \rangle$$

$$(\textsc{true})\ \frac{}{\mathtt{true} \Rightarrow \mathtt{true}} \qquad (\textsc{false})\ \frac{}{\mathtt{false} \Rightarrow \mathtt{false}}$$

$$(\textsc{if-t})\ \frac{e \Rightarrow \mathtt{true} \quad e_1 \Rightarrow v}{\mathtt{if}\ e\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2 \Rightarrow v} \qquad (\textsc{if-f})\ \frac{e \Rightarrow \mathtt{false} \quad e_2 \Rightarrow v}{\mathtt{if}\ e\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2 \Rightarrow v}$$

$$(\textsc{t-var})\ \frac{}{\Gamma \vdash x : T} \quad \Gamma(x) = T \qquad (\textsc{t-bool})\ \frac{}{\Gamma \vdash b : \mathtt{Bool}} \quad b \in \{\mathtt{true}, \mathtt{false}\}$$

$$(\textsc{t-fld})\ \frac{\Gamma \vdash e : \mathsf{C}}{\Gamma \vdash e.\mathsf{f}_i : T_i} \quad \begin{array}{l}\mathsf{fields}(\mathsf{C}) = T_1\,\mathsf{f}_1\,; \ldots T_n\,\mathsf{f}_n\,; \\ i \in 1..n\end{array}$$

$$(\textsc{t-new})\ \frac{\Gamma \vdash e_i : T_i \quad \forall i \in 1..n}{\Gamma \vdash \mathsf{new}\ \mathsf{C}(e_1, \ldots, e_n) : \mathsf{C}} \quad \mathsf{fields}(\mathsf{C}) = T_1\,\mathsf{f}_1\,; \ldots T_n\,\mathsf{f}_n\,;$$

$$(\textsc{t-invk})\ \frac{\Gamma \vdash e : \mathsf{C} \quad \Gamma \vdash e_i : T_i \quad \forall i \in 1..n}{\Gamma \vdash e.\mathsf{m}(e_1, \ldots, e_n) : T} \quad \mathsf{mtype}(\mathsf{C}, \mathsf{m}) = T_1 \ldots T_n \to T$$

$$(\textsc{t-if})\ \frac{\Gamma \vdash e : \mathtt{Bool} \quad \Gamma \vdash e_1 : T \quad \Gamma \vdash e_2 : T}{\Gamma \vdash \mathtt{if}\ e\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2 : T} \qquad (\textsc{t-sub})\ \frac{\Gamma \vdash e : T}{\Gamma \vdash e : T'} \quad T <: T'$$

$$(\textsc{t-}\vee\textsc{-elim})\ \frac{\Gamma \vdash e : \bigvee_{i \in 1..m} \mathsf{C}_i \quad \Gamma, x{:}\mathsf{C}_i \vdash E[x] : T \quad \forall i \in 1..m}{\Gamma \vdash E[e] : T} \quad x \text{ fresh}$$

FIGURE 5.12 MINIFJ$^{\vee}$: syntax, big-step semantics and type system

```
class CorD {
  C∨D el;
  A getf() { this.el.f }
  CorD update(A x) { new CorD(this.el.update(x)) }
}
```

Bodies of methods `getf` and `update` in class `CorD` are well-typed thanks to rule (T-∨-ELIM), as shown by the following derivation for `update`, where Γ = x:A, this:CorD.

$$\dfrac{\Gamma \vdash \texttt{this.el} : \texttt{C} \vee \texttt{D} \qquad \dfrac{\Gamma, y{:}\texttt{C} \vdash y.\texttt{update}(x) : \texttt{C}}{\Gamma, y{:}\texttt{C} \vdash y.\texttt{update}(x) : \texttt{C} \vee \texttt{D}} \qquad \dfrac{\Gamma, y{:}\texttt{D} \vdash y.\texttt{update}(x) : \texttt{D}}{\Gamma, y{:}\texttt{D} \vdash y.\texttt{update}(x) : \texttt{C} \vee \texttt{D}}}{\dfrac{\Gamma \vdash \texttt{this.el.update}(x) : \texttt{C} \vee \texttt{D}}{\Gamma \vdash \texttt{new CorD}(\texttt{this.el.update}(x)) : \texttt{CorD}}}$$

The above example can be typed in FJ∨ as well, even though with a different technique.[6] On the other hand, with our more uniform approach inspired by rule (∨E), we can type examples where the same subexpression having a union type occurs more than once, and soundness relies on the determinism of evaluation, as in the example at the beginning of this section.

To illustrate this, let us consider an example. Consider the expression e = `if false then new C(...) else new D(...)`, given the above class table. By rule (T-IF), the expression e has type C∨D, and, by rule (T-∨-ELIM), the expression e`.eq(`e`)` has type `Bool`, as shown by the following derivation:

$$\dfrac{\vdash e : \texttt{C} \vee \texttt{D} \qquad x{:}\texttt{C} \vdash x.\texttt{eq}(x) : \texttt{Bool} \quad x{:}\texttt{D} \vdash x.\texttt{eq}(x) : \texttt{Bool}}{\vdash e.\texttt{eq}(e) : \texttt{Bool}}$$

This expression cannot be typed in FJ∨, because there is no way to eliminate the union type assigned to e when it occurs as an argument.

Quite surprisingly, subject reduction fails for the expected small-step semantics, even if there are no intersection types, which are the source, together with the (∨E) rules, of the problems in the λ-calculus. Indeed, we have the following small-step reduction:

$$e.\texttt{eq}(e) \longrightarrow \texttt{new D}(\ldots).\texttt{eq}(e) \longrightarrow \texttt{new D}(\ldots).\texttt{eq}(\texttt{new D}(\ldots))$$

where the intermediate expression cannot be typed, because e has a union type. This happens because intersection types are in a sense hidden in the class table: the method eq occurs in two different classes with different types, hence, roughly, we could assign it the intersection type $(\texttt{C}\,\texttt{C} \to \texttt{Bool}) \wedge (\texttt{D}\,\texttt{D} \to \texttt{Bool})$.

As in previous examples, the soundness proof uses an inversion lemma and a substitution lemma. The canonical forms lemma is trivial since the only values of type C are objects (constructor calls with values as arguments) instances of a subclass. In addition, we need a lemma (dubbed "key") which assures that a value typed by a union of classes can also be typed by one of these classes. The proof of this lemma is straightforward, since values having class types are just new constructors, as shown by canonical forms.

6 When the receiver of a method call has a union type, look-up (function mtype) is directly performed and gives a set of method signatures; arguments should comply all parameter types and the type of the call is the union of return types.

LEMMA 5.53 (Substitution): If $\Gamma\{T'/x\} \vdash e : T$ and $\Gamma \vdash e' : T'$, then $\Gamma \vdash e[e'/x] : T'$.

LEMMA 5.54 (Canonical forms): The following hold:

1. If $\Gamma \vdash v : \mathtt{Bool}$, then $v = \mathtt{true}$ or $v = \mathtt{false}$.
2. If $\Gamma \vdash v : \mathtt{C}$, then $v = \mathtt{new}\ \mathtt{D}(v_1, \ldots, v_n)$ and $\mathtt{D} <: \mathtt{C}$.

LEMMA 5.55 (Inversion): The following hold:

1. If $\Gamma \vdash x : T$, then $\Gamma(x) <: T$.
2. If $\Gamma \vdash e.\mathsf{f} : T$, then $\Gamma \vdash e : \bigvee_{i \in 1..m} \mathtt{C}_i$ and, for all $i \in 1..m$, $\mathsf{fields}(\mathtt{C}_i) = T_{i1}\,\mathsf{f}_{i1}\,; \ldots T_{in_i}\,\mathsf{f}_{in_i}\,;$ and $\mathsf{f} = \mathsf{f}_{ik_i}$ and $T_{ik_i} <: T$ for some $k_i \in 1..n_i$.
3. If $\Gamma \vdash \mathtt{new}\ \mathtt{C}(e_1, \ldots, e_n) : T$, then $\mathtt{C} <: T$ and $\mathsf{fields}(\mathtt{C}) = T_1\,\mathsf{f}_1\,; \ldots T_n\,\mathsf{f}_n\,;$ and $\Gamma \vdash e_i : T_i$ for all $i \in 1..n$.
4. If $\Gamma \vdash e_0.\mathsf{m}(e_1, \ldots, e_n) : T$, then $\Gamma \vdash e_0 : \bigvee_{i \in 1..m} \mathtt{C}_i$ and, there is $p \in 0..n$ such that $e_0 = \ldots = e_p$ and, for all $i \in 1..m$,
   - $\mathsf{mtype}(\mathtt{C}_i, \mathsf{m}) = T_{i1} \ldots T_{in} \to T_i$, and
   - for all $k \in 1..p$, $\mathtt{C}_i <: T_{ik}$, and
   - for all $k \in p+1..n$, $\Gamma \vdash e_k : T_{ik}$, and
   - $T_i <: T$.
5. If $\Gamma \vdash \mathtt{if}\ e\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2 : T$, then $\Gamma \vdash e : \mathtt{Bool}$ and $\Gamma \vdash e_1 : T$ and $\Gamma \vdash e_2 : T$.

Proof sketch We prove only points 2 and 4.

2. The proof is by induction on the derivation of $\Gamma \vdash e.\mathsf{f} : T$. For rule (T-FLD), we have $\Gamma \vdash e : \mathtt{C}$, $\mathsf{fields}(\mathtt{C}) = T_1\,\mathsf{f}_1\,; \ldots T_n\,\mathsf{f}_n\,;$, $\mathsf{f}_i = \mathsf{f}$ and $T_i = T$, for some $i \in 1..n$. For rule (T-SUB), the thesis is immediate by induction hypothesis. For rule (T-V-ELIM), we have $E = [].\mathsf{f}$, $\Gamma \vdash e : \bigvee_{i \in 1..m} \mathtt{C}_i$ and $\Gamma, x{:}\mathtt{C}_i \vdash E[x] : T$, for all $i \in 1..m$, then, by induction hypothesis, for all $i \in 1..m$, we get $\Gamma, x{:}\mathtt{C}_i \vdash x : \bigvee_{j \in 1..m_i} \mathtt{D}_{ij}$ and, for all $j \in 1..m_i$, $\mathsf{fields}(\mathtt{D}_{ij}) = T_{j11}\,\mathsf{f}_{j1}\,; \ldots T_{jn_j}\,\mathsf{f}_{jn_j}\,;$ and $T_{jk_j} <: T$, for some $k_j \in 1..n_j$. Since $\Gamma, x{:}\mathtt{C}_i \vdash x : \bigvee_{j \in 1..m_j} \mathtt{D}_{ij}$, we have $\mathtt{C}_i <: \bigvee_{j \in 1..m_j} \mathtt{D}_{ij}$, hence $\mathtt{C}_i <: \mathtt{D}_{ij_i}$, for some $j_i \in 1..m_i$, by definition of subtyping. Then the thesis follows easily by assumption (FJ2).
4. The proof is by induction on the derivation of $\Gamma \vdash e_0.\mathsf{m}(e_1, \ldots, e_n) : T$. For rule (T-INVK), we have $\Gamma \vdash e_0 : \mathtt{C}_0, p = 0, \mathsf{mtype}(\mathtt{C}_0, \mathsf{m}) = T_1 \ldots T_n \to T$, and, for all $k \in 1..n$, $\Gamma \vdash e_k : T_k$. For rule (T-SUB), the thesis is immediate by induction hypothesis. For rule (T-V-ELIM), we have $E = [].\mathsf{m}([], \ldots, [], e_{p+1}, \ldots, e_n)$, hence p is the number of holes in E and $e_0 = \ldots = e_p$, and $\Gamma \vdash e_0 : \bigvee_{i \in 1..m} \mathtt{C}_i$ and, for all $i \in 1..m$, $\Gamma, x{:}\mathtt{C}_i \vdash E[x] : T$, with x fresh. By induction hypothesis, we know that, for all $i \in 1..m$, $\Gamma, x{:}\mathtt{C}_i \vdash x : \bigvee_{j \in 1..m_i} \mathtt{D}_{ij}$ and there is $p_i \in 1..n$ such that the first p_i argu-

ments of $E[x]$ are equal to the receiver, namely x and this implies $p_i \leq p$ because x is fresh. Let $i \in 1..m$. Since $\Gamma, x{:}\mathsf{C}_i \vdash x : \bigvee_{j\in 1..m_i} \mathsf{D}_{ij}$, we get $\mathsf{C}_i <: \bigvee_{j\in 1..m_j} \mathsf{D}_{ij}$, thus $\mathsf{C}_i <: \mathsf{D}_{ij_i}$, for some $j_i \in 1..m_i$, by definition of subtyping. Therefore, by induction hypothesis and assumption (FJ3), we get $\mathsf{mtype}(\mathsf{C}_i, \mathsf{m}) = T_{i1} \ldots T_{in} \to T_i$ and, for all $k \in 1..p_i$, $\mathsf{D}_{ij_i} <: T_{ik}$, hence $\mathsf{C}_i <: T_{ik}$, and, for all $k \in p_i + 1..p$, $\Gamma, x{:}\mathsf{C}_i \vdash x : T_{ik}$, hence $\mathsf{C}_i <: T_{ik}$ and, for all $k \in p + 1..n$, $\Gamma, x{:}\mathsf{C}_i \vdash e_k : T_{ik}$, hence, because x does not occur in e_k as it is fresh, by contraction we get $\Gamma \vdash e_k : T_{ik}$, and, finally, $T_i <: T$.

□

LEMMA 5.56 (Key): If $\Gamma \vdash v : \bigvee_{1\leq i\leq n} \mathsf{C}_i$, then $\Gamma \vdash v : \mathsf{C}_i$ for some $i \in 1 \ldots n$.

In order to state soundness, let $\langle C_4, R_4, \mathcal{R}_4 \rangle$ be the big-step semantics defined in Figure 5.12 (C_4 is the set of expressions and R_4 is the set of values), and let $\Pi 4^C_T = \{e \in C_4 \mid\vdash e : T\}$ and $\Pi 4^R_T = \{v \in R_4 \mid\vdash v : T\}$, for T defined in Figure 5.12. We need a last lemma to prove soundness:

LEMMA 5.57 (Determinism): If $\mathcal{R}_4 \vdash_\mu e \Rightarrow v_1$ and $\mathcal{R}_4 \vdash_\mu e \Rightarrow v_2$, then $v_1 = v_2$.

Proof: Straightforward induction on rules in $\mathcal{R}_4$, because every syntactic construct has a unique big-step meta-rule. □

THEOREM 5.58 (Soundness): The big-step semantics $\langle C_4, R_4, \mathcal{R}_4 \rangle$ and the indexed predicate $\Pi 4$ satisfy the conditions (LP), ($\exists$P) and ($\forall$P) of Section 5.5.2.

Proof sketch We sketch the proof only of (LP) for rule (INVK), other cases and conditions are similar to previous proofs.

For rule (INVK), Lemma 5.55 (4) applied to $\vdash e_0.\mathsf{m}(e_1, \ldots, e_n) : T$ implies $\vdash e_0 : \bigvee_{i\in 1..m} \mathsf{C}_i$ and, there is $p \in 0..n$ such that $e_0 = \ldots = e_p$ and, for all $i \in 1..m$, $\mathsf{mtype}(\mathsf{C}_i, \mathsf{m}) = T_{i1} \ldots T_{in} \to T_i$, and for all $k \in 1..p$, $\mathsf{C}_i <: T_{ik}$, and for all $k \in p + 1..n$, $\vdash e_i : T_{ik}$, and $T_i <: T$. Assuming $\vdash \mathsf{new}\ \mathsf{C}(vs) : \bigvee_{i\in 1..m} \mathsf{C}_i$, by Lemma 5.56 and Lemma 5.54, we get $\mathsf{C} <: \mathsf{C}_i$ for some $i \in 1..m$. Since $\mathsf{mtype}(\mathsf{C}_i, \mathsf{m}) = T_{i1} \ldots T_{in} \to T_i$ and $\mathsf{mbody}(\mathsf{C}, \mathsf{m}) = \langle x_1 \ldots x_n, e\rangle$, by assumption (FJ3) and (FJ1), $\mathtt{this}{:}\mathsf{C}, x_1{:}T_{i1}, \ldots, x_n{:}T_{in} \vdash e : T_i$. Assume, for all $k \in 1..p$, $\vdash v_k : \bigvee_{i\in 1..m} \mathsf{C}_i$ and, for all $k \in p + 1..n$, $\vdash v_k : T_{ik}$, then, since $e_0 = \ldots = e_p$, by Lemma 5.57, we get $v_1 = \ldots = v_p = \mathsf{new}\ \mathsf{C}(vs)$, hence $\vdash v_k : T_{ik}$, for all $k \in 1..p$, because $\mathsf{C} <: \mathsf{C}_i <: T_{ik}$ for all $k \in 1..p$. Lemma 5.53 gives $\vdash e[v_1/x_1] \ldots [v_n/x_n][\mathsf{new}\ \mathsf{C}(vs)/\mathtt{this}] : T_i$. Finally, we can conclude $\vdash v : T$ by rule (T-SUB), as $T_i <: T$. □

5.6.5 Imperative FJ

This last example shows how our technique behaves in an imperative setting. In Figure 5.13 and Figure 5.14 we show a minimal imperative extension of

$$
\begin{array}{lcll}
e & ::= & x \mid e.\mathsf{f} \mid \mathsf{new}\ \mathsf{C}(e_1,\ldots,e_n) \mid e.\mathsf{m}(e_1,\ldots,e_n) \mid e.\mathsf{f}{=}e' \mid \iota & \text{expressions} \\
c & ::= & \langle \mu, e\rangle & \text{configurations} \\
r & ::= & \langle \mu, \iota\rangle & \text{results}
\end{array}
$$

$$
(\textsc{obj})\ \frac{}{\langle \mu, \iota\rangle \Rightarrow \langle \mu, \iota\rangle} \qquad (\textsc{fld})\ \frac{\langle \mu, e\rangle \Rightarrow \langle \mu', \iota\rangle}{\langle \mu, e.\mathsf{f}_i\rangle \Rightarrow \langle \mu', \iota_i\rangle}\ \begin{array}{l}\mu'(\iota) = \mathsf{new}\ \mathsf{C}(\iota_1,\ldots,\iota_n) \\ \mathsf{fields}(\mathsf{C}) = \mathsf{C}_1\ \mathsf{f}_1\mathsf{;}\ \ldots\ \mathsf{C}_n\ \mathsf{f}_n\mathsf{;} \\ i \in 1..n\end{array}
$$

$$
(\textsc{new})\ \frac{\langle \mu_i, e_i\rangle \Rightarrow \langle \mu_{i+1}, \iota_i\rangle \quad \forall i \in 1..n}{\langle \mu, \mathsf{new}\ \mathsf{C}(e_1,\ldots,e_n)\rangle \Rightarrow \langle \mu', \iota\rangle}\ \begin{array}{l}\mu_1 = \mu \\ \mu' = \mu_{n+1}\{\mathsf{new}\ \mathsf{C}(\iota_1,\ldots,\iota_n)/\iota\} \\ \iota\ \text{fresh}\end{array}
$$

$$
(\textsc{invk})\ \frac{\begin{array}{l}\langle \mu_i, e_i\rangle \Rightarrow \langle \mu_{i+1}, \iota_i\rangle \quad \forall i \in 0..n \\ \langle \mu_{n+1}, e[\iota_1/x_1]\ldots[\iota_n/x_n][\iota_0/\mathtt{this}]\rangle \Rightarrow \langle \mu', \iota\rangle\end{array}}{\langle \mu, e_0.\mathsf{m}(e_1,\ldots,e_n)\rangle \Rightarrow \langle \mu', \iota\rangle}\ \begin{array}{l}\mu_0 = \mu \\ \mu_1(\iota_0) = \mathsf{new}\ \mathsf{C}(\_) \\ \mathsf{mbody}(\mathsf{C},\mathsf{m}) = \langle x_1\ldots x_n, e\rangle \\ \mu' = \mu_{n+1}\end{array}
$$

$$
(\textsc{fld-up})\ \frac{\langle \mu, e\rangle \Rightarrow \langle \mu', \iota\rangle \quad \langle \mu', e'\rangle \Rightarrow \langle \mu'', \iota'\rangle}{\langle \mu, e.\mathsf{f}_i{=}e'\rangle \Rightarrow \langle \mu''_{[\iota.i=\iota']}, \iota'\rangle}\ \begin{array}{l}\mu(\iota) = \mathsf{new}\ \mathsf{C}(\iota_1,\ldots,\iota_n) \\ \mathsf{fields}(\mathsf{C}) = \mathsf{C}_1\ \mathsf{f}_1\mathsf{;}\ \ldots\ \mathsf{C}_n\ \mathsf{f}_n\mathsf{;} \\ i \in 1..n\end{array}
$$

FIGURE 5.13 Imperative FJ: syntax and big-step semantics

FJ. We assume a well-typed class table and we use the notations introduced in Section 5.6.2. Expressions are enriched with field assignment and *object identifiers* ι, which only occur in runtime expressions. A *memory* μ maps object identifiers to *object states*, which are expressions of shape $\mathsf{new}\ \mathsf{C}(\iota_1,\ldots\iota_n)$. Results are configurations of shape $\langle \mu, \iota\rangle$. We denote by $\mu_{[\iota.i=\iota']}$ the memory obtained from μ by replacing by ι' the i-th field of the object state associated with ι. The *type assignment* Σ maps object identifiers into types (class names). We write $\Sigma \vdash e : \mathsf{C}$ for $\emptyset; \Sigma \vdash e : \mathsf{C}$.

As for the other examples, to prove soundness we need some standard properties of the typing rules: inversion and substitution lemmas.

LEMMA 5.59 (Inversion): The following hold:

1. If $\Gamma; \Sigma \vdash \langle \mu, e\rangle : \mathsf{C}$, then $\Gamma; \Sigma \vdash \mu(\iota) : \Sigma(\iota)$ for all $\iota \in \mathrm{dom}(\mu)$ and $\Sigma \vdash e : \mathsf{C}$ and $\mathrm{dom}(\Sigma) = \mathrm{dom}(\mu)$.
2. If $\Gamma; \Sigma \vdash x : \mathsf{C}$, then $\Gamma(x) <: \mathsf{C}$.
3. If $\Gamma; \Sigma \vdash e.\mathsf{f}_i : \mathsf{C}$, then $\Gamma; \Sigma \vdash e : \mathsf{D}$ and $\mathsf{fields}(\mathsf{D}) = \mathsf{C}_1\ \mathsf{f}_1\mathsf{;}\ \ldots\ \mathsf{C}_n\ \mathsf{f}_n\mathsf{;}$ and $\mathsf{C}_i <: \mathsf{C}$ where $i \in 1..n$.
4. If $\Gamma; \Sigma \vdash \mathsf{new}\ \mathsf{C}(e_1,\ldots,e_n) : \mathsf{D}$, then $\mathsf{C} <: \mathsf{D}$ and $\mathsf{fields}(\mathsf{C}) = \mathsf{C}_1\ \mathsf{f}_1\mathsf{;}\ \ldots\ \mathsf{C}_n\ \mathsf{f}_n\mathsf{;}$ and $\Gamma; \Sigma \vdash e_i : \mathsf{C}_i$ for all $i \in 1..n$.
5. If $\Gamma; \Sigma \vdash e_0.\mathsf{m}(e_1,\ldots,e_n) : \mathsf{C}$, then $\Gamma; \Sigma \vdash e_i : \mathsf{C}_i$ for all $i \in 0..n$ and $\mathsf{mtype}(\mathsf{C}_0,\mathsf{m}) = \mathsf{C}_1\ldots\mathsf{C}_n \to \mathsf{D}$ with $\mathsf{D} <: \mathsf{C}$.
6. If $\Gamma; \Sigma \vdash e.\mathsf{f}_i{=}e' : \mathsf{C}$, then $\Gamma; \Sigma \vdash e : \mathsf{D}$ and $\mathsf{fields}(\mathsf{D}) = \mathsf{C}_1\ \mathsf{f}_1\mathsf{;}\ \ldots\ \mathsf{C}_n\ \mathsf{f}_n\mathsf{;}$ and $\Gamma; \Sigma \vdash e' : \mathsf{C}_i$ and $\mathsf{C}_i <: \mathsf{C}$.

$$(\text{T-CONF})\ \frac{\Sigma \vdash \mu(\iota) : \Sigma(\iota)\ \forall \iota \in \mathsf{dom}(\mu) \quad \Sigma \vdash e : \mathsf{C}}{\Sigma \vdash \langle \mu, e \rangle : \mathsf{C}} \quad \mathsf{dom}(\Sigma) = \mathsf{dom}(\mu)$$

$$(\text{T-VAR})\ \frac{}{\Gamma; \Sigma \vdash x : \mathsf{C}} \quad \Gamma(x) = \mathsf{C}$$

$$(\text{T-FLD})\ \frac{\Gamma; \Sigma \vdash e : \mathsf{C} \quad \mathsf{fields}(\mathsf{C}) = \mathsf{C}_1\, \mathsf{f}_1\, ; \ldots \mathsf{C}_n\, \mathsf{f}_n\, ;}{\Gamma; \Sigma \vdash e.\mathsf{f}_i : \mathsf{C}_i \quad i \in 1..n}$$

$$(\text{T-NEW})\ \frac{\Gamma; \Sigma \vdash e_i : \mathsf{C}_i \quad \forall i \in 1..n}{\Gamma; \Sigma \vdash \mathsf{new}\ \mathsf{C}(e_1, \ldots, e_n) : \mathsf{C}} \quad \mathsf{fields}(\mathsf{C}) = \mathsf{C}_1\, \mathsf{f}_1\, ; \ldots \mathsf{C}_n\, \mathsf{f}_n\, ;$$

$$(\text{T-INVK})\ \frac{\Gamma; \Sigma \vdash e_i : \mathsf{C}_i \quad \forall i \in 0..n}{\Gamma; \Sigma \vdash e_0.\mathsf{m}(e_1, \ldots, e_n) : \mathsf{C}} \quad \mathsf{mtype}(\mathsf{C}_0, \mathsf{m}) = \mathsf{C}_1 \ldots \mathsf{C}_n \to \mathsf{C}$$

$$(\text{T-FLD-UP})\ \frac{\begin{array}{l}\Gamma; \Sigma \vdash e : \mathsf{C} \\ \Gamma; \Sigma \vdash e' : \mathsf{C}_i\end{array} \quad \begin{array}{l}\mathsf{fields}(\mathsf{C}) = \mathsf{C}_1\, \mathsf{f}_1\, ; \ldots \mathsf{C}_n\, \mathsf{f}_n\, ; \\ i \in 1..n\end{array}}{\Gamma; \Sigma \vdash e.\mathsf{f}_i{=}e' : \mathsf{C}_i}$$

$$(\text{T-OID})\ \frac{}{\Gamma; \Sigma \vdash \iota : \mathsf{C}} \quad \Sigma(\iota) = \mathsf{C} \qquad (\text{T-SUB})\ \frac{\Gamma; \Sigma \vdash e : \mathsf{C}}{\Gamma; \Sigma \vdash e : \mathsf{C}'} \quad \mathsf{C} <: \mathsf{C}'$$

FIGURE 5.14 Imperative FJ: typing rules

7. If $\Gamma; \Sigma \vdash \iota : \mathsf{C}$, then $\Sigma(\iota) <: \mathsf{C}$.

LEMMA 5.60 (Substitution): If $\Gamma\{\mathsf{C}'/x\}; \Sigma \vdash e : \mathsf{C}$ and $\Gamma; \Sigma \vdash e' : \mathsf{C}'$, then $\Gamma; \Sigma \vdash e[e'/x] : \mathsf{C}$.

Let $\langle C_5, R_5, \mathcal{R}_5 \rangle$ be the big-step semantics defined in Figure 5.13. We can prove the soundness of the indexed predicate $\Pi 5$ defined by: $\Pi 5^{C}_{\langle \Sigma, \mathsf{C} \rangle} = \{\langle \mu, e \rangle) \in C_5 \mid \Sigma' \vdash \langle \mu, e \rangle : \mathsf{C}$ for some Σ' s.t. $\Sigma \subseteq \Sigma'\}$ and $\Pi 5^{R}_{\langle \Sigma, \mathsf{C} \rangle} = R_5 \cap \Pi 5^{C}_{\langle \Sigma, \mathsf{C} \rangle}$. The type assignment Σ' is needed, since memory can grow during evaluation.

THEOREM 5.61 (Soundness): The big-step semantics $\langle C_5, R_5, \mathcal{R}_5 \rangle$ and the indexed predicate $\Pi 5$ satisfy the conditions (LP), ($\exists$P) and ($\forall$P) of Section 5.5.2.

Proof: We prove separately the three conditions

PROOF OF (LP): The proof is by cases on instantiations of meta-rules.

Case: (OBJ) Trivial from the hypothesis.

Case: (FLD) Lemma 5.59 (1) applied to $\Sigma \vdash \langle \mu, e.\mathsf{f}_i \rangle : \mathsf{C}$ implies $\Sigma \vdash \mu(\iota) : \Sigma(\iota)$ for all $\iota \in \mathsf{dom}(\mu)$ and $\Sigma \vdash e.\mathsf{f}_i : \mathsf{C}$ and $\mathsf{dom}(\Sigma) = \mathsf{dom}(\mu)$. Lemma 5.59 (3) applied to $\Sigma \vdash e.\mathsf{f}_i : \mathsf{C}$ implies $\Sigma \vdash e : \mathsf{D}$ and $\mathsf{fields}(\mathsf{D}) = \mathsf{C}_1\, \mathsf{f}_1\, ; \ldots \mathsf{C}_n\, \mathsf{f}_n\, ;$ and $\mathsf{C}_i <: \mathsf{C}$ where $i \in 1..n$. Since $\langle \mu, e \rangle \Rightarrow \langle \mu', \iota \rangle$ is a premise we assume $\Sigma' \vdash \langle \mu', \iota \rangle : \mathsf{D}$ with $\Sigma \subseteq \Sigma'$. Lemma 5.59 (1) and Lemma 5.59 (7) imply $\Sigma'(\iota) <: \mathsf{D}$. Lemma 5.59 (4) allows us to get $\mu'(\iota) = \mathsf{new}\ \mathsf{C}'(\iota_1, \ldots \iota_m)$ with $n \leq m$ and $\mathsf{C}' <: \mathsf{D}$ and $\Sigma' \vdash \iota_i : \mathsf{C}_i$. So we conclude $\Sigma' \vdash \langle \mu', \iota_i \rangle : \mathsf{C}$ by rules (T-SUB) and (T-CONF).

Case: (NEW) Lemma 5.59 (1) applied to $\Sigma \vdash \langle \mu, \mathsf{new\ C}(e_1, \ldots, e_n)\rangle : \mathsf{D}$ implies $\Sigma \vdash \mu(\iota) : \Sigma(\iota)$ for all $\iota \in \mathrm{dom}(\mu)$ and $\Sigma \vdash \mathsf{new\ C}(e_1, \ldots, e_n) : \mathsf{D}$ and $\mathrm{dom}(\Sigma) = \mathrm{dom}(\mu)$. Lemma 5.59 (4) applied to $\Sigma \vdash \mathsf{new\ C}(e_1, \ldots, e_n) : \mathsf{D}$ implies $\mathsf{C} <: \mathsf{D}$ and $\mathsf{fields(C)} = \mathsf{C}_1\,\mathsf{f}_1\,; \ldots \mathsf{C}_n\,\mathsf{f}_n\,;$ and $\Sigma \vdash e_i : \mathsf{C}_i$ for all $i \in 1..n$. Since $\langle \mu, e_i\rangle \Rightarrow \langle \mu_{i+1}, \iota_i\rangle$ is a premise we assume $\Sigma_i \vdash \langle \mu_{i+1}, \iota_i\rangle : \mathsf{C}_i$ for all $i \in 1..n$ with $\Sigma \subseteq \Sigma_1 \subseteq \cdots \subseteq \Sigma_n$. Lemma 5.59 (1) and Lemma 5.59 (7) imply $\Sigma_i(\iota_i) <: \mathsf{C}_i$ for all $i \in 1..n$. Using rules (T-OID), (T-NEW) and (T-SUB) we derive $\Sigma_n \vdash \mathsf{new\ C}(\iota_1, \ldots, \iota_n) : \mathsf{D}$. We then conclude $\Sigma_n, \iota : \mathsf{D} \vdash \langle \mu_{n+1}, \iota\rangle : \mathsf{D}$ by rules (T-OID)and (T-CONF).

Case: (INVK) Lemma 5.59 (1) applied to $\Sigma_0 \vdash \langle \mu_0, e_0\mathsf{.m}(e_1, \ldots, e_n)\rangle : \mathsf{C}$ implies $\Sigma_0 \vdash \mu_0(\iota) : \Sigma_0(\iota)$ for all $\iota \in \mathrm{dom}(\mu_0)$ and $\Sigma_0 \vdash e_0\mathsf{.m}(e_1, \ldots, e_n) : \mathsf{C}$ and $\mathrm{dom}(\Sigma_0) = \mathrm{dom}(\mu_0)$. Lemma 5.59 (5) applied to $\Sigma_0 \vdash e_0\mathsf{.m}(e_1, \ldots, e_n) : \mathsf{C}$ implies $\Sigma_i \vdash e_i : \mathsf{C}_i$ for all $i \in 0..n$ and $\mathsf{mtype}(\mathsf{C}_0, \mathsf{m}) = \mathsf{C}_1 \ldots \mathsf{C}_n \to \mathsf{D}$ with $\mathsf{D} <: \mathsf{C}$. Since $\langle \mu_i, e_i\rangle \Rightarrow \langle \mu_{i+1}, \iota_i\rangle$ is a premise we assume $\Sigma_i \vdash \langle \mu_{i+1}, \iota_i\rangle : \mathsf{C}_i$ for all $i \in 0..n$ with $\Sigma_0 \subseteq \cdots \subseteq \Sigma_n$. Lemma 5.59 (1) gives $\Sigma_i \vdash \iota_i : \mathsf{C}_i$ for all $i \in 0..n$. The typing of the class table implies $x_1{:}\mathsf{C}_1, \ldots, x_n{:}\mathsf{C}_n, \mathtt{this}{:}\mathsf{C}_0 \vdash e : \mathsf{D}$. Lemma 5.60 gives $\Sigma_n \vdash e' : \mathsf{D}$ where $e' = e[\iota_1/x_1] \ldots [\iota_n/x_n][\iota_0/\mathtt{this}]$. Using rules (T-SUB) and (T-CONF) we derive $\Sigma_n \vdash \langle \mu_{n+1}, e'\rangle : \mathsf{C}$. Since $\langle \mu_{n+1}, e'\rangle \Rightarrow \langle \mu', \iota\rangle$ is a premise we conclude $\Sigma' \vdash \langle \mu', \iota\rangle : \mathsf{C}$ with $\Sigma_n \subseteq \Sigma'$.

Case: (FLD-UP) Lemma 5.59 (1) applied to $\Sigma \vdash \langle \mu, e\mathsf{.f}_i{=}e'\rangle : \mathsf{C}$ implies $\Sigma \vdash \mu(\iota) : \Sigma(\iota)$ for all $\iota \in \mathrm{dom}(\mu)$ and $\Sigma \vdash e\mathsf{.f}_i{=}e' : \mathsf{C}$ and $\mathrm{dom}(\Sigma) = \mathrm{dom}(\mu)$. Lemma 5.59 (6) applied to $\Sigma \vdash e\mathsf{.f}_i{=}e' : \mathsf{C}$ implies $\Sigma \vdash e : \mathsf{D}$ and $\mathsf{fields(D)} = \mathsf{C}_1\,\mathsf{f}_1\,; \ldots \mathsf{C}_n\,\mathsf{f}_n\,;$ and $\Sigma \vdash e' : \mathsf{C}_i$ and $\mathsf{C}_i <: \mathsf{C}$. Since $\langle \mu, e\rangle \Rightarrow \langle \mu', \iota\rangle$ and $\langle \mu', e'\rangle \Rightarrow \langle \mu'', \iota'\rangle$ are premises we assume $\Sigma' \vdash \langle \mu', \iota\rangle : \mathsf{D}$ and $\Sigma'' \vdash \langle \mu'', \iota'\rangle : \mathsf{C}_i$, with $\Sigma \subseteq \Sigma' \subseteq \Sigma''$. Notice that $\mu''(\iota)$ and $\mu''_{[\iota.i=\iota']}(\iota)$ have the same types for all ι by construction. We conclude $\Sigma'' \vdash \langle \mu''_{[\iota.i=\iota']}, \iota'\rangle : \mathsf{C}_i$.

PROOF OF ($\exists$P): All the closed expressions appear as conclusions in the reduction rules.

PROOF OF ($\forall$P): Since the only values are configurations with object identifiers it is easy to verify that the premises of the reduction rules are satisfied, being the conditions on memory and object identifiers assured by the typing rules. □

6

Big-step semantics with observations

As discussed in Chapter 5, the behaviour of programs or software systems can be described by the final *results* of computations. However, in many cases, this provides only a partial description of such behaviour, because programs and systems can also interact with the external environment. For instance, a function call can terminate and return a value, as well as have output effects during its execution. Hence, to provide a richer description of the behaviour of programs and systems, we should take into account these interactions as well, also seen as *observations* made during the computations.

In this chapter, we deal with (operational) semantic definitions covering both results and observations. Often, such definitions are provided for *finite* computations only. Notably, in big-step style, as discussed in detail in Chapter 5, infinite computations are simply not modelled, hence diverging and stuck terms are not distinguished. This becomes even more unsatisfactory if we have observations, since non-terminating programs can exhibit a significant observable behaviour interacting with the context, even though they do not produce any final result, and we would like to be able to model such a situation.

As shown in Section 5.4, inference systems with corules can be successfully adopted to express big-step semantics modelling diverging computations, where corules play an essential role to control coinduction. Indeed, modeling infinite behaviour by a purely coinductive interpretation of big-step rules would lead to spurious results and undetermined observations, as discussed by Leroy and Grall (2009), Ancona (2012, 2014), and Ancona, Dagnino, and Zucca (2017c, 2018), whereas, by adding appropriate corules, we can correctly get divergence (∞) as the only result, and a uniquely determined observation. This approach has been adopted by Ancona, Dagnino, and Zucca (2017c, 2018) to design big-step definitions including infinite behaviour for λ-calculus and a simple imperative Java-like language. However, in such works the designer of the semantics is in charge of finding the appropriate corules, and this is a non-trivial task.

In this chapter, as already done in Chapter 5 for semantics without observations, we show a construction that extends a given big-step semantics, modeling finite computations, to include infinite behaviour as well, notably *generating appropriate corules*. The construction consists of two steps:

1. Starting from a monoid O modeling finite observations (*e.g.*, finite traces),

we construct an *ω-monoid* $\langle O, O_\infty \rangle$ also modeling infinite observations (*e.g.*, infinite traces). The latter structure is a variation of ω-semigroup (Perrin and Pin, 2004), including a *mixed product* composing a finite with a possibly infinite observation, and an *infinite product* mapping an infinite sequence of finite observations into a single (possibly infinite) one.

2. Starting from an inference system defining a big-step judgement $c \Rightarrow \langle r, o \rangle$, with c configuration, $r \in R$ result, and $o \in O$ finite observation, we construct an inference system with corules defining an extended big-step judgment $c \Rightarrow \langle r_\infty, o_\infty \rangle$ with $r_\infty \in R_\infty = R + \{\infty\}$ and $o_\infty \in O_\infty$. The construction generates additional rules for *propagating divergence*, as in Section 5.4, and corules for *introducing divergence* in a controlled way, obtained as instances of two patterns (CO-UNIT) and (CO-GEN).

To show the effectiveness of our approach, we provide several instances of the framework, with different kinds of (finite) observations. Depending on the nature of such observations, instantiations of only (CO-UNIT) or both should be added to obtain the intended infinite behaviour.

Finally, we consider the issue of formally justifying that the construction is correct. To this end, we extend the approach considered in Chapter 5 to take into account observations: given a big step semantics, we define a labelled transition relation, modelling the evaluation algorithm guided by rules, hence we can model computations, as usual, by sequences of transition steps and then, the observation produced by a computation is the possibly infinite product of all the observations labelling single steps. Therefore, to prove correctness of our construction, we just have to prove that the resulting semantics is equivalent to that obtained from the labelled transition relation. This proof of equivalence holds for deterministic semantics; issues arising in the non-deterministic case and a possible solution are sketched in the Chapter 7.

The chapter is organised as follows. Section 6.1 informally introduces our approach on a simple example. Section 6.2 describes the construction of ω-monoids, and Section 6.3 defines big-step semantics with observations, the labelled transition relations modelling computation steps and the extension of big-step semantics. Section 6.4 treats several significant examples and Section 6.5 contains the proof of correctness.

6.1 An introductory example

AN EXAMPLE OF SEMANTICS WITH OBSERVATIONS We illustrate our approach on an example discussed by Ancona, Dagnino, and Zucca (2018): a call-by-value λ-calculus with output.

The top section of Figure 6.1 contains the syntax. We assume infinite sets of *variables x* and *integer constants n*. Results, namely values, are either integer constants or λ-abstractions. Beyond standard constructs, we add expressions of shape `out` *e*, which output the result of the evaluation of *e*. Correspondingly, observations are sequences of such outputs, and the semantics of an expression

$$
\begin{array}{rcll}
e & ::= & v \mid x \mid e_1\, e_2 \mid \mathtt{out}\, e & \text{expressions} \\
u, v & ::= & n \mid \lambda x.e & \text{values} \\
o & ::= & v_1 \ldots v_n & \text{finite observations}
\end{array}
$$

$$
(\textsc{val})\ \frac{}{v \Rightarrow \langle v, \varepsilon \rangle} \qquad (\textsc{out})\ \frac{e \Rightarrow \langle v, o \rangle}{\mathtt{out}\, e \Rightarrow \langle v, o \cdot v \rangle}
$$

$$
(\textsc{app})\ \frac{e_1 \Rightarrow \langle \lambda x.e, o_1 \rangle \quad e_2 \Rightarrow \langle v_2, o_2 \rangle \quad e[v_2/x] \Rightarrow \langle v, o \rangle}{e_1\, e_2 \Rightarrow \langle v, o_1 \cdot o_2 \cdot o \rangle}
$$

FIGURE 6.1 λ-calculus with output: syntax and finite semantics

consists of both its final result and the whole observation produced during the computation.

The bottom section contains big-step rules defining the operational semantics of the language. As usual, the big-step judgement $e \Rightarrow \langle v, o \rangle$ *directly* computes the semantics (result and observation) of the expression.

EXTENDING OBSERVATIONS First of all we enrich results by a special element ∞ denoting divergence, and observations by considering infinite output sequences:

$$
\begin{array}{rcll}
v_\infty & ::= & v \mid \infty & \text{results or divergence} \\
o_\infty & ::= & o \mid v_1 \ldots v_n \ldots & \text{observations}
\end{array}
$$

The latter is an instance of a general construction, formally defined in Section 6.2. Briefly, assuming that finite observations are a *monoid* $\langle O, *, \mathsf{e} \rangle$, with $*$ (sequentially) combining two observations, and e the identity, also called *unit*, modeling absence of observation, we construct an *ω-monoid* $\langle O, O_\infty \rangle$, where O_∞ models possibily infinite observations, with a *mixed product* $*^{\mathsf{m}} : O \times O_\infty \to O_\infty$ combining a finite with a possibly infinite observation, and an *infinite product* $\mathsf{p} : O^\omega \to O_\infty$ mapping an infinite sequence of finite observations into a possibly infinite observation. For details and a proper definition, see Section 6.2.

In the example, the monoid is $\langle \mathsf{Val}^\star, \cdot, \varepsilon \rangle$, and the construction just adds infinite output sequences (Val is the set of values). Formally, we obtain the ω-monoid $\langle \mathsf{Val}^\star, \mathsf{Val}^\infty \rangle$, where the mixed product is the concatenation of a finite with a possibly infinite sequence, still denoted by $\cdot$, and the infinite product returns the concatenation of an infinite number of finite sequences.

EXTENDING BIG-STEP SEMANTICS We modify the judgement into $e \Rightarrow \langle v_\infty, o_\infty \rangle$ to include divergence and infinite observations. Correspondingly, we extend the inference system, as will be formalized in Section 6.3. Here we informally explain the extension using the example.

DIVERGENCE PROPAGATION We first present the easier part, which is how to add rules for divergence propagation, shown in Figure 6.2.

These rules are not arbitrary: they are constructed in a systematic manner starting from the original (meta-)rules. That is, for each original

$$\text{(DIV-APP1)}\ \frac{e_1 \Rightarrow \langle \infty, o_\infty \rangle}{e_1\ e_2 \Rightarrow \langle \infty, o_\infty \rangle} \qquad \text{(DIV-APP2)}\ \frac{e_1 \Rightarrow \langle \lambda x.e, o \rangle \quad e_2 \Rightarrow \langle \infty, o_\infty \rangle}{e_1\ e_2 \Rightarrow \langle \infty, o\ \cdot\ o_\infty \rangle}$$

$$\text{(DIV-APP3)}\ \frac{e_1 \Rightarrow \langle \lambda x.e, o_1 \rangle \quad e_2 \Rightarrow \langle v_2, o_2 \rangle \quad e[v_2/x] \Rightarrow \langle \infty, o_\infty \rangle}{e_1\ e_2 \Rightarrow \langle \infty, o_1 \cdot o_2 \cdot o_\infty \rangle}$$

$$\text{(DIV-OUT)}\ \frac{e \Rightarrow \langle \infty, o_\infty \rangle}{\texttt{out}\ e \Rightarrow \langle \infty, o_\infty \rangle}$$

FIGURE 6.2 λ-calculus with output: adding divergence propagation

Infinite proof tree for any $\Omega \Rightarrow \langle v_\infty, o_\infty \rangle$

$$\dfrac{\dfrac{}{\omega \Rightarrow \langle \omega, \varepsilon \rangle} \quad \dfrac{}{\omega \Rightarrow \langle \omega, \varepsilon \rangle} \quad \dfrac{\vdots}{\omega\ \omega = (x\ x)[\omega/x] \Rightarrow \langle v_\infty, o_\infty \rangle}}{\Omega = \omega\ \omega \Rightarrow \langle v_\infty, \varepsilon \cdot \varepsilon \cdot o_\infty = o_\infty \rangle}$$

Finite proof tree with corules for $\Omega \equiv \omega\ \omega \Rightarrow \langle \infty, \varepsilon \rangle$

$$\text{(CO-EMPTY)}\ \frac{}{\Omega \Rightarrow \langle \infty, \varepsilon \rangle}$$

FIGURE 6.3 Proof trees for Ω

meta-rule, we consider premises as ordered from left to right. For each premise, say, the i-th, we add a meta-rule where the first $i - 1$ premises are kept as they are (hence, the corresponding computations converge), whereas the i-th premise requires the corresponding computation to diverge. In the conclusion, we get ∞ as result and the mixed product of the observations in the premises (in the given order) as observation; only the last observation is possibly infinite.

DIVERGENCE INTRODUCTION The rules in Figure 6.2 ensure that divergent computations, if any, are correctly propagated. To discuss how to correctly *introduce* divergent computations, consider, for instance, the term $\Omega = \omega\,\omega$, where $\omega = \lambda x.x\,x$. We should derive $\Omega \Rightarrow \langle \infty, \varepsilon \rangle$, and *only this* judgment, modeling that Ω diverges without producing any output. Similarly to what happens without observations (cf. Section 5.4), no judgment can be derived for Ω in the inductive interpretation of rules, and, in the coinductive interpretation, an infinite proof tree exists for *any* judgment $\Omega \Rightarrow \langle v_\infty, o_\infty \rangle$, as shown in Figure 6.3, where we apply either (APP), if v_∞ is a value v, or (DIV-APP3), if $v_\infty = \infty$.

In summary, divergent terms have no result (are stuck) in the inductive interpretation, and a fully non-deterministic result in the coinductive interpretation. Our approach is to add appropriate corules, so that, as in Section 5.4, we add constraints to filter out wrong judgments.

In the example, we add to the rules in Figure 6.1 and Figure 6.2 the corules shown in Figure 6.4, which again are obtained in a systematic manner.

$$\text{(CO-EMPTY)}\ \frac{}{e \Rightarrow \langle \infty, \varepsilon \rangle} \qquad \text{(CO-OUT)}\ \frac{e \Rightarrow \langle v, o \rangle}{\mathtt{out}\, e \Rightarrow \langle \infty, o \cdot v \cdot o_\infty \rangle}$$

FIGURE 6.4 λ-calculus with output: adding corules

Notably, they are special cases of two patterns, named (CO-UNIT) and (CO-GEN), which handle two different cases of divergent computations. Here we explain the role of these rules; they will be formally defined in Section 6.3 (Definition 6.32).

The (CO-UNIT) pattern handles the case where the computation produces a *finite*[1] observation o. In this case, a purely coinductive approach obtains any v_∞, and any observation of shape $o *^{\mathsf{m}} o_\infty$, and the aim of the corule is to only allow $v_\infty = \infty$ and $o_\infty = \mathsf{e}$. In the example, we use the specific name (CO-EMPTY), since the unit is the empty sequence. In the Ω case, $o = \varepsilon$, and, with (CO-EMPTY), we derive only the judgment $\Omega \Rightarrow \langle \infty, \varepsilon \rangle$. Indeed, consider one of the proof trees in Figure 6.3, which have an infinite path. For each node of such an infinite path[2] the corules should allow a finite proof tree. If the path consists of infinitely many nodes $\Omega \Rightarrow \langle v, o_\infty \rangle$, for some v and o_∞, then the corules do not allow any finite proof tree for this judgment, since they all have ∞ in the conclusion. If it consists of infinite nodes $\Omega \Rightarrow \langle \infty, o_\infty \rangle$, for some o_∞, then it is easy to see that only for $\Omega \Rightarrow \langle \infty, \varepsilon \rangle$ there is a finite proof tree, shown in the bottom section of Figure 6.3.

The (CO-GEN) pattern, instead, handles the case where the computation produces an infinite observation, since *infinitely many elementary non-unit observations are produced.* In this case, a purely coinductive approach obtains any v_∞; on the other side, the observation is uniquely determined by this infinite sequence. Consider, for instance, the term $\hat{\Omega} = \hat{\omega}\,(\mathtt{out}\,\hat{\omega})$, with $\hat{\omega} = \lambda x.(x\,(\mathtt{out}\,x))$, which is expected to diverge producing the output sequence consisting of infinitely many occurrences of the value $\hat{\omega}$. In the top part of Figure 6.5 we show the infinite proof trees which can be constructed for $\hat{\Omega}$. Each of them forces the constraint $o_\infty = \hat{\omega} \cdot o_\infty$, which is solved only for $\hat{o}_\infty = \hat{\omega} \ldots \hat{\omega} \ldots$. Hence, the aim of the corule is, on one hand, to force $v_\infty = \infty$, and, on the other hand, to allow a finite proof tree for any node in the infinite path. Since in this infinite path there are infinite nodes producing an observation, it is enough to add a corule for such nodes. In our running example, we use the specific name (CO-OUT), since the only original meta-rule producing a (non-unit) observation is (OUT).

Exactly as in the Ω case, the corules do not allow finite proof trees for judgments of shape $\hat{\Omega} \Rightarrow \langle v, o_\infty \rangle$. On the other hand, they should allow a finite proof tree for the judgment $\hat{\Omega} \Rightarrow \langle \infty, \hat{o}_\infty \rangle$, which can be obtained by corule (CO-OUT), as shown in the bottom section of Figure 6.5.

1 More precisely, *finitely generated* by elementary observations, as defined in Definition 6.13.
2 For other nodes the condition is true since they have a finite proof tree using the rules.

Infinite proof tree for $\hat{\Omega} \Rightarrow \langle v_\infty, o_\infty \rangle$, enforcing $o_\infty = \hat{\omega} \cdot o_\infty$

$$\dfrac{\dfrac{}{\hat{\omega} \Rightarrow \langle \hat{\omega}, \varepsilon \rangle} \quad \dfrac{\dfrac{}{\hat{\omega} \Rightarrow \langle \hat{\omega}, \varepsilon \rangle}}{\mathtt{out}\,\hat{\omega} \Rightarrow \langle \hat{\omega}, \hat{\omega} \rangle} \quad \dfrac{\vdots}{\hat{\Omega} = \hat{\omega}\,(\mathtt{out}\,\hat{\omega}) = (x\,(\mathtt{out}\,x))[\hat{\omega}/x] \Rightarrow \langle v_\infty, o_\infty \rangle}}{\hat{\Omega} = \hat{\omega}\,(\mathtt{out}\,\hat{\omega}) \Rightarrow \langle v_\infty, \varepsilon \cdot \hat{\omega} \cdot o_\infty = \hat{\omega} \cdot o_\infty \rangle}$$

Finite proof tree with corules for $\hat{\Omega} \Rightarrow \langle \infty, \hat{o}_\infty \rangle$ with $\hat{o}_\infty = \hat{\omega} \ldots \hat{\omega} \ldots$

$$\dfrac{\dfrac{}{\hat{\omega} \Rightarrow \langle \hat{\omega}, \varepsilon \rangle} \quad \dfrac{\dfrac{}{\hat{\omega} \Rightarrow \langle \hat{\omega}, \varepsilon \rangle}}{\mathtt{out}\,\hat{\omega} \Rightarrow \langle \infty, \hat{\omega} \cdot \hat{o}_\infty = \hat{o}_\infty \rangle}}{\hat{\Omega} = \hat{\omega}\,(\mathtt{out}\,\hat{\omega}) \Rightarrow \langle \infty, \hat{o}_\infty \rangle}$$

FIGURE 6.5 Proof trees for $\hat{\Omega}$

We conclude by explaining how meta-corules are added in a systematic way.

- We always add a meta-coaxiom (CO-UNIT) with conclusion $e \Rightarrow \langle \infty, \mathsf{e} \rangle$.
- Assuming that in each meta-rule the observation in the conclusion is the product $o'_0 * o_1 * o'_1 * \cdots * o_n * o'_n$, where $o_1, \ldots, o_n$ are the observations in the premises, and $o'_0, \ldots, o'_n$ are elementary observations produced by such meta-rule, we add, for each meta-rule and $i \in 0..n$ where $o'_i \neq \mathsf{e}$, a corresponding meta-corule with the first $i-1$ premises and conclusion $e \Rightarrow \langle \infty, o'_0 * o_1 * o'_1 * \cdots * o_{i-1} * o'_{i-1} *^{\mathsf{m}} o_\infty \rangle$. In the example, only (OUT) has a non-unit elementary observation, therefore (CO-OUT) is the only added meta-corule.

A formal account of this general construction is given in Section 6.3.

6.2 From finite to infinite observations

In this section, we formally define ω-monoids. They are a variation of ω-semigroups used in algebraic language theory (Perrin and Pin, 2004). Further, we introduce a completion construction from monoids to ω-monoids and finally, as a digression, we analyse its properties in categorical terms.

6.2.1 ω-monoids

In this subsection we will define ω-monoids. The definition is a straightforward extension of that of ω-semigroups, see, *e.g.*, (Perrin and Pin, 2004), to take into account the identity of the monoid. We start by recalling basic definitions about monoids.

DEFINITION 6.1 : A *monoid* is a triple $\langle M, *, \mathsf{e}\rangle$ where M is a set, $* : M \times M \to M$ is an associative binary operation and $\mathsf{e} \in M$ is an *identity* (a.k.a. neutral element or unit), that is, we have $x * \mathsf{e} = x = \mathsf{e} * x$, for all $x \in M$.

A *monoid homomorphism* from $\langle M, *_M, \mathsf{e}_M\rangle$ into $\langle N, *_N, \mathsf{e}_N\rangle$ is a function $f : M \to N$ such that $f(x *_M y) = f(x) *_N f(y)$ and $f(\mathsf{e}_M) = \mathsf{e}_N$.

As it is common practice, when there is no confusion, we will denote a monoid $\langle M, *, \mathsf{e}\rangle$ just by its underlying set M.

Given a set A, recall that $A^\star$ is (the underlying set of) a *free monoid*, where the product is given by concatenation and the identity by the empty sequence. This means that, if M is a monoid, then for every map $f : A \to M$ there is a unique monoid homomorphism $f^\sharp : A^\star \to M$ such that $f^\sharp(a) = f(a)$, for all $a \in A$. In particular, starting from the identity $\mathsf{id}_M : M \to M$, we get the map $\mathsf{id}_M^\sharp : M^\star \to M$, which interprets a sequence of elements of M as a unique element, by *iterating* the operation $*$ on the sequence. We abbreviate $\mathsf{id}_M^\sharp$ by it_M (for "iterator"), dropping the subscript when clear from the context.

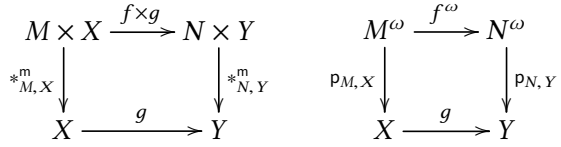
To define ω-monoids, we need to introduce for a monoid $\langle M, *, \mathsf{e}\rangle$ a relation on M^ω, used to state the infinite associative law, the distinguishing axiom of ω-semigroups and ω-monoids. Given an infinite sequence $\sigma \in M^\omega$, a *decomposition* of σ is a sequence $(u_i)_{i\in\mathbb{N}}$ of non-empty finite sequences ($u_i \in M^+$), such that σ can be obtained by flattening $(u_i)_{i\in\mathbb{N}}$, that is, $\sigma = u_0u_1u_2\ldots$. Then, for all $\sigma, \tau \in M^\omega$, we will write $\sigma \bowtie \tau$ iff there are a decomposition $(u_i)_{i\in\mathbb{N}}$ of σ and a decomposition $(v_i)_{i\in\mathbb{N}}$ of τ such that for all $i \in \mathbb{N}$, $\mathsf{it}(u_i) = \mathsf{it}(v_i)$. The relation $\bowtie$ can equivalently be characterised coinductively, as the greatest fixed point of the monotone function $P(R) = \{\langle u\sigma, v\tau\rangle \in M^\omega \times M^\omega \mid u, v \in M^+,\ \mathsf{it}(u) = \mathsf{it}(v),\ \langle\sigma, \tau\rangle \in R\}$ on the lattice of relations on M^ω, ordered by inclusion.

DEFINITION 6.2 (ω-monoid): An *ω-monoid* is a pair $\langle M, X\rangle$ of sets together with a function $* : M \times M \to M$, called *finite product*, a function $*^{\mathsf{m}} : M \times X \to X$, called *mixed product*, a function $\mathsf{p} : M^\omega \to X$, called *infinite product*, and a constant $\mathsf{e} \in M$, called *identity* (a.k.a. unit), satisfying the following properties:

1. $\langle M, *, \mathsf{e}\rangle$ is a monoid (cf. Definition 6.1),
2. $*^{\mathsf{m}}$ is a left action: for all $x, y \in M$ and $z \in X$: $x *^{\mathsf{m}} (y *^{\mathsf{m}} z) = (x * y) *^{\mathsf{m}} z$ and $\mathsf{e} *^{\mathsf{m}} z = z$,
3. p respects the mixed product: for all $x \in M$ and $\sigma \in M^\omega$, $x *^{\mathsf{m}} \mathsf{p}(\sigma) = \mathsf{p}(x\sigma)$,
4. p satisfies the infinite associative law: for all $\sigma, \tau \in M^\omega$, if $\sigma \bowtie \tau$, then $\mathsf{p}(\sigma)$=$\mathsf{p}(\tau)$.

An *ω-monoid homomorphism* from $\langle M, X\rangle$ to $\langle N, Y\rangle$ is a pair of functions $f : M \to N$ and $g : X \to Y$ such that f is a monoid homomorphism and the

following diagrams commute:



EXAMPLE 6.3 : We list a few basic examples of ω-monoids.

1. A main example is the pair $\langle A^\star, A^\infty\rangle$ of finite sequences and possibly infinite sequences over an alphabet A. Finite and infinite products are given by concatenation. The mixed product concatenates finite sequences (on the left) with arbitrary sequences (on the right). There is no concatenation with infinite sequences on the left.
2. As a special case of Item 1 (when A is a singleton set), $\langle \mathbb{N}, +, 0\rangle$ extends to the ω-monoid $\langle \mathbb{N}, \mathbb{N} + \{\infty\}\rangle$.
3. The monoid $\langle \mathbb{N}, \vee, 0\rangle$ (where $n_1 \vee n_2$ is the join of n_1 and n_2 w.r.t. the standard order) also extends to an ω-monoid $\langle \mathbb{N}, \mathbb{N} + \{\infty\}\rangle$. Here, the infinite product computes the supremum of values occuring in a sequence.
4. Let $\wp(X)$ be the powerset of a set X, and $\wp_\omega(X)$ the finite powerset, i.e., $\wp_\omega(X) = \{S \subseteq X \mid S \text{ finite}\}$. The monoid $\langle \wp_\omega(X), \cup, \emptyset\rangle$ extends to an ω-monoid, with the second component given by the (full) powerset $\wp(X)$.

REMARK: The last requirement in Definition 6.2 cannot be derived from the previous ones: take $\langle \mathbb{Z}, \mathbb{Z}_\omega\rangle$ where $\mathbb{Z}_\omega = \mathbb{Z} \cup \{\pm\infty, \bot\}$, where p computes the sum of the elements of an infinite sequence σ, returning $\bot$ if it is undetermined. Then, the sum of $\sigma = 1{:}-1{:}1{:}-1{:}\ldots$ is undetermined, hence $\mathsf{p}(\sigma) = \bot$. However, for $\tau = 0{:}0{:}0{:}\ldots$, we have $\tau \bowtie \sigma$ and $\mathsf{p}(\tau) = 0 \neq \bot = \mathsf{p}(\sigma)$.

The following result, which will be proved using categorical tools in Section 6.2.3, characterises free ω-monoids generated by a set A.

PROPOSITION 6.4 (Free ω-monoid): For every set A, ω-monoid $\langle M, X\rangle$ and map $f : A \to M$, there is a unique ω-monoid homomorphism $\langle f^\sharp, f^\sharp_\infty\rangle : \langle A^\star, A^\infty\rangle \to \langle M, X\rangle$ such that $f^\sharp(a) = f(a)$, for all $a \in A$.

The map $f^\sharp_\infty$ in the above statement is given explicitly as the composition

$$f^\sharp_\infty = \left(A^\infty \xrightarrow{f^\infty} M^\infty \xrightarrow{j} M^\omega \xrightarrow{\mathsf{p}} X\right)$$

where p is the infinite product of the ω-monoid $\langle M, X\rangle$ and j is the identity on infinite sequences, and maps a finite sequence u to $u\mathsf{e}^\omega$, with e the unit of M.

Analogously to the monoid case, we abbreviate $\langle f^\sharp, f^\sharp_\infty\rangle$ by $\langle \mathsf{it}, \mathsf{it}^\infty\rangle$ when $f = \mathsf{id}_M$. The function it^∞ allows to interpret a possibly infinite sequence of elements of M as a unique element of X, intuitively multiplying all the elements of the sequence.

6.2.2 Left continuous monoids and completion

We present now a construction which, given a monoid $\langle M, *, \mathsf{e}\rangle$, produces an ω-monoid $C_\infty(M) = \langle M, M_\infty\rangle$ called the *completion* of M. The idea behind the completion is to construct the infinite product of a sequence $\sigma = (x_i)_{i\in\mathbb{N}}$ as some kind of "limit" of the sequence of its finite approximations $x_0 * \cdots * x_i$, for all $i \in \mathbb{N}$. The completion we define is order-theoretic, and follows essentially from the chain completion presented by Markowsky (1976).

The starting point of the construction is the observation that any monoid carries an implicit (pre)order structure, which in a sense, is based on how much information each element carries. Let $\langle M, *, \mathsf{e}\rangle$ be a monoid and define a binary relation $\preceq_*$ on M as follows: for all $x, y \in M$, $x \preceq_* y$ iff $y = x * z$ for some $z \in M$. It is easy to check that this relation is reflexive and transitive, hence it is a preorder, and multiplication on the left preserves the relation, that is, the left multiplication function $\mathsf{m}_x : M \to M$, defined by $\mathsf{m}_x(y) = x * y$, is monotone. Furthermore, the identity e is the least element with respect to such preorder, that is, $\mathsf{e} \preceq_* x$, for all $x \in M$, and this implies that $x \preceq_* x * y$, for all $x, y \in M$; intuitively, the identity models "absence of information" and multiplying means "adding information". Finally, any monoid homomorphism $f : M \to N$ is monotone with respect to such preorders on M and N.

EXAMPLE 6.5 : In the monoid $\langle A^\star, \cdot, \varepsilon\rangle$ (finite sequences and concatenation), the relation $\preceq_*$ is the standard prefix order. In the monoids $\langle \mathbb{N}, +, 0\rangle$ and $\langle \mathbb{R}, +, 0\rangle$ (natural and real numbers with addition), the relation $\preceq_*$ is the standard linear order, as in the monoid $\langle \mathbb{N}, \vee, 0\rangle$. In the monoid $\langle \mathbb{N}, \cdot, 1\rangle$ (natural numbers with multiplication), the relation $\preceq_*$ is the standard division order. In the monoid $\langle \wp(X), \cup, \emptyset\rangle$ (subsets of X with union), the relation $\preceq_*$ is the standard set inclusion.

Similarly, given an ω-monoid $\langle M, X\rangle$, we can define a relation $\preceq_{*^\mathsf{m}}$ from M to X as follows: for all $x \in M$ and $z \in X$, $x \preceq_{*^\mathsf{m}} y$ iff $y = x *^\mathsf{m} z$ for some $z \in X$. Now, given a sequence $\sigma = (x_n)_{n\in\mathbb{N}}$, for all $n \in \mathbb{N}$ we denote by $\sigma[n]$ the prefix[3] $x_0 \dots x_{n-1}$ of σ. Then, the sequence of *partial products* of σ, $(\mathsf{it}(\sigma[n]))_{n\in\mathbb{N}}$, is increasing with respect to $\preceq_*$, because $\mathsf{it}(\sigma[n]) \preceq_* \mathsf{it}(\sigma[n]) * x_n = \mathsf{it}(\sigma[n+1])$.

DEFINITION 6.6 : Let $\langle M, X\rangle$ be an ω-monoid and $\sigma = (x_n)_{n\in\mathbb{N}} \in M^\omega$. A *limit product* of σ in X is an element $z \in X$ such that there is a sequence $(z_n)_{n\in\mathbb{N}}$ where $z_0 = z$ and, for all $n \in \mathbb{N}$, $z_n \in X$ and $z_n = x_n *^\mathsf{m} z_{n+1}$.

In other words, we can associate with $\sigma = (x_n)_{n\in\mathbb{N}}$ a system of equations $(Z_n{=}x_n{*}Z_{n+1})_{n\in\mathbb{N}}$, where $(Z_n)_{n\in\mathbb{N}}$ is an infinite sequence of variables; then, the sequence $(z_n)_{n\in\mathbb{N}}$ of elements in X is a solution of such system of equations and a limit product of σ is the value of the variable Z_0. Note that, by construction, we have $z = x_0 * \cdots x_{n-1} * z_n = \mathsf{it}(\sigma[n]) *^\mathsf{m} z_n$, for all $n \in \mathbb{N}$, hence $\mathsf{it}(\sigma[n]) \preceq_{*^\mathsf{m}} z$, that is, z is an upper bound of the sequence of partial products of σ, or, alternatively, each partial product of σ is a "finite" approximation of z.

3 Note that, when $n = 0$, $\sigma[n] = \varepsilon$.

The interesting fact is that the infinite product of a sequence in an ω-monoid is always a limit product, as shown by the next proposition:

PROPOSITION 6.7 : Let $\langle M, X\rangle$ be an ω-monoid and $\sigma \in M^\omega$, then $\mathsf{p}(\sigma)$ is a limit product of σ in X.

Proof: Let $\sigma = (x_i)_{i\in\mathbb{N}}$ and $\sigma_n = (x_{n+i})_{i\in\mathbb{N}}$, for all $n \in \mathbb{N}$, hence $\sigma_0 = \sigma$, and, for all $n \in \mathbb{N}$, $\sigma_n = x_n\sigma_{n+1}$. Then, setting $z_n = \mathsf{p}(\sigma_n)$, we have $z_0 = \mathsf{p}(\sigma)$ and, by Definition 6.2, $z_n = \mathsf{p}(\sigma_n) = x_n *^{\mathsf{m}} \mathsf{p}(\sigma_{n+1}) = x_n *^{\mathsf{m}} z_{n+1}$, as needed. □

Therefore, these observations point out that, in order to construct an infinite product of a sequence of elements in M, we can construct a (least) upper bound of the sequence of partial products, and this is the strategy we will follow.

The completion construction deals with a special class of monoids defined below. Given an increasing sequence $(x_n)_{n\in\mathbb{N}}$ in M, we denote by $\sup_{n\in\mathbb{N}} x_n$ its supremum[4], if any.

DEFINITION 6.8 (Left continuous monoids): A *left continuous monoid* is a monoid $\langle M, *, \mathsf{e}\rangle$ such that

- $\preceq_*$ is a partial order and
- for all $x \in M$ and increasing chain $(y_n)_{n\in\mathbb{N}}$ in M, $x * \sup_{n\in\mathbb{N}} y_n = \sup_{n\in\mathbb{N}}(x * y_n)$.

A *continuous homomorphism* from $\langle M, *_M, \mathsf{e}_M\rangle$ to $\langle N, *_N, \mathsf{e}_N\rangle$ is a monoid homomorphism $f : \langle M, *_M, \mathsf{e}_M\rangle \to \langle N, *_N, \mathsf{e}_N\rangle$ such that, for any increasing sequence $(x_n)_{n\in\mathbb{N}}$ in M, $f(\sup_{n\in\mathbb{N}} x_n) = \sup_{n\in\mathbb{N}} f(x_n)$.

In other words, for a left continuous monoid $\langle M, *, \mathsf{e}\rangle$, we require $\preceq_*$ to be antisymmetric, and, for all $x \in M$, the left multiplication function m_x to be continuous, that is, to preserve suprema of increasing sequences, which are unique as $\preceq_*$ is antisymmetric.

The fact that $\preceq_*$ is a partial order, that is, it satisfies the antisymmetric property, can be characterised algebraically as follows:

PROPOSITION 6.9 : Let M be a monoid, $\preceq_*$ is a partial order iff, for all $x, y, z \in M$, $x * y * z = x$ implies $x * y = x$.

Proof: The preorder $\preceq_*$ is antisymmetric iff , for all $w, x, y, z \in M$, $w = x * y$ and $x = w * z$ implies $x = w$, iff, for all $x, y, z \in M$, $x = x * y * z$ implies $x = x * y$. □

Therefore, every monoid satisfying the above condition, like those in Example 6.3, can be endowed by a partial order, namely $\preceq_*$, derived from its binary operation, whose bottom element is the identity of the monoid. Note

4 This is not unique in general, as $\preceq_*$ is not antisymmetric, but we will use this notation only when the supremum is unique, namely, when $\preceq_*$ is a partial order.

that a side effect of the condition in Proposition 6.9 is that the only invertible element is the identity, hence, for instance, the preorder $\preceq_*$ on a group, such as integers $\mathbb{Z}$ with the addition, is not a partial order. Finally, it is easy to check that, in all the examples in Example 6.3, left multiplication functions are continuous with respect ot $\preceq_*$, hence they are left continuous monoids.

The reason why we define our construction on left continuous monoids is that we want to look at infinite products as suprema of certain chains and so the completion construction has to take into account existing suprema. For instance, when completing the powerset $\wp(X)$ of a set X (cf. Example 6.3), we would like to obtain no new elements because all suprema already exist.

The completion construction below turns a left continuous monoid M into an ω-monoid $C_\infty(M) = \langle M, M_\infty\rangle$, where M_∞ is presented as a quotient of the set M^ω.

Assume a left continuous monoid $\langle M, \preceq_*, *, \mathsf{e}\rangle$. We start by defining a relation $\sqsubseteq$ on M^ω. Let $\sigma \in M^\omega$, we write $\sigma[n]$ for the prefix of length n of σ, that is, if $\sigma = (x_i)_{i\in\mathbb{N}}$, then $\sigma[n] = x_0 \dots x_{n-1}$. We define the set $\mathsf{S}(\sigma) \subseteq M$ as the closure under suprema of increasing chain of the set $P_\sigma = \{\mathsf{it}(\sigma[n]) \mid n \in \mathbb{N}\}$. More explicitly, since all non-stationary[5] increasing chains in P_σ are subchains of $(\mathsf{it}(\sigma[n]))_{n\in\mathbb{N}}$, we can characterise $\mathsf{S}(\sigma)$ as follows:

$$\mathsf{S}(\sigma) = \{\mathsf{it}(\sigma[n]) \mid n \in \mathbb{N}\} \cup \{\sup_{n\in\mathbb{N}} \mathsf{it}(\sigma[n])\}$$

that is, $\mathsf{S}(\sigma)$ is the set of products of all prefixes of σ plus their supremum, if any. Then, for all $\sigma, \tau \in M^\omega$, we define

$$\sigma \sqsubseteq \tau \Leftrightarrow \forall x \in \mathsf{S}(\sigma).\ \exists y \in \mathsf{S}(\tau).\ x \preceq_* y$$

This relation is a preorder. We denote by $\equiv$ the induced equivalence relation, that is, $\sigma \equiv \tau$ iff $\sigma \sqsubseteq \tau$ and $\tau \sqsubseteq \sigma$.

DEFINITION 6.10 (Completion): The *completion* of a left continuous monoid $\langle M, \preceq_*, *, \mathsf{e}\rangle$ is the ω-monoid $C_\infty(M) = \langle M, M_\infty\rangle$ where:

- $M_\infty = M^\omega/\equiv$,
- the mixed product $*^{\mathsf{m}} : M \times M_\infty \to M_\infty$ is given by $x *^{\mathsf{m}} [\tau]_\equiv = [x\tau]_\equiv$, and
- the infinite product $\mathsf{p} : M^\omega \to M_\infty$ is given by $\mathsf{p}(\tau) = [\tau]_\equiv$.

The fact that $C_\infty(M)$ is indeed an ω-monoid follows from the next lemma:

LEMMA 6.11 : Let $\langle M, \preceq_*, *, \mathsf{e}\rangle$ be a left continuous monoid. The following hold:

1. for all $z \in M$ and $\sigma, \tau \in M^\omega$, if $\sigma \equiv \tau$, then $z\sigma \equiv z\tau$,
2. for all $\sigma, \tau \in M^\omega$, if $\sigma \bowtie \tau$ then $\sigma \equiv \tau$,
3. for all $z_1, z_2 \in M$, $z_1\mathsf{e}^\omega \equiv z_2\mathsf{e}^\omega$ iff $z_1 = z_2$,

5 A sequence $(x_n)_{n\in\mathbb{N}}$ is stationary when there is $k \in \mathbb{N}$ such that $x_n = x_k$ for all $n \geq k$.

4. for any left continuous monoid homomorphism $f : M \to N$ and $\sigma, \tau \in M^\omega$, if $\sigma \equiv \tau$, then $f^\omega(\sigma) \equiv f^\omega(\tau)$.

Proof:

1. We have to prove the two inequalities $z\sigma \sqsubseteq z\tau$ and $z\tau \sqsubseteq z\sigma$; we prove only the first, as the other is analogous.

   Let $x \in \mathsf{S}(z\sigma)$, then $x = \sup_{i\in\mathbb{N}}(\mathsf{it}(u_i))$ for some increasing sequence $(u_i)_i$ of prefixes of $z : \sigma$, i.e., u_i is a prefix of $z\sigma$, for all i. Without loss of generality, assume that $(u_i)_i$ does not contain the empty word (since $\mathsf{it}(\varepsilon)$ is the least element in M and the case where the supremum is the identity is trivial) so each u_i is of the form $u_i = zu_i'$, with u_i' a prefix of σ. Then $\sup_{i\in\mathbb{N}}(\mathsf{it}(u_i')) \in S(\sigma)$, thus, since $\sigma \sqsubseteq \tau$, there is an increasing sequence $(v_i)_i$ with each v_i a prefix of τ, and $\sup_{i\in\mathbb{N}}(\mathsf{it}(u_i')) \preceq_* \sup_{i\in\mathbb{N}}(\mathsf{it}(v_i))$. So we get

$$\begin{aligned} x &= \sup_{i\in\mathbb{N}}(\mathsf{it}(u_i)) \\ &= \sup_{i\in\mathbb{N}}(\mathsf{it}(zu_i')) \\ &= \sup_{i\in\mathbb{N}}(z * \mathsf{it}(u_i')) \\ &= z * \sup_{i\in\mathbb{N}}(\mathsf{it}(u_i')) \\ &\preceq_* z * \sup_{i\in\mathbb{N}}(\mathsf{it}(v_i)) \\ &= \sup_{i\in\mathbb{N}}(\mathsf{it}(zv_i)) \end{aligned}$$

   and since $sup_{i\in\mathbb{N}}(\mathsf{it}(zv_i)) \in S(z\tau)$ we get $z\sigma \sqsubseteq z\tau$, as needed.

2. Suppose $\sigma \bowtie \tau$, with $\sigma = (x_i)_{i\in\mathbb{N}}$ and $\tau = (y_i)_{i\in\mathbb{N}}$, *i.e.*, there are decompositions $(u_i)_{i\in\mathbb{N}}$ and $(v_i)_{i\in\mathbb{N}}$ of σ and τ, respectively, such that, for all $i \in \mathbb{N}$, $\mathsf{it}(u_i) = \mathsf{it}(v_i)$. Towards a proof of $\sigma \sqsubseteq \tau$, let $x \in S(\sigma)$. We have two cases:

   - If $x = \mathsf{it}(u)$ for some prefix u of σ and $(u_i)_{i\in\mathbb{N}}$ is a decomposition of σ, then there is $n \in \mathbb{N}$ such that u is a prefix of $\hat{u} = u_0 \dots u_n$, hence $x = \mathsf{it}(u) \preceq_* \mathsf{it}(\hat{u})$. By hypothesis, we have $\mathsf{it}(\hat{u}) = \mathsf{it}(u_0 \dots u_n) = \mathsf{it}(v_0 \dots v_n) \in \mathsf{S}(\tau)$ and this proves the thesis.
   - If $x = \sup_{i\in\mathbb{N}}(\mathsf{it}(\sigma[i])$, then note that, as $(u_i)_{i\in\mathbb{N}}$ is a decomposition of σ, we have that, for all $i \in \mathbb{N}$, $\sigma[i]$ is a prefix of $\hat{u}_i = u_0 \dots u_{n_i}$ for some $n_i \in \mathbb{N}$, hence $\mathsf{it}(\sigma[i]) \preceq_* \mathsf{it}(\hat{u}_i)$. By hypothesis, we get $\mathsf{it}(\hat{u}_i) = \mathsf{it}(v_0 \dots v_{n_i}) \preceq_* \sup_{i\in\mathbb{N}}(\mathsf{it}(\tau[i]))$, and this implies $x \preceq_* \sup_{i\in\mathbb{N}}(\mathsf{it}(\tau[i])) \in \mathsf{S}(\tau)$, as needed.

   The other inequality $\tau \sqsubseteq \sigma$ follows from what we have just proved because $\bowtie$ is symmetric.

3. Suppose $z_1\mathsf{e}^\omega \equiv z_2\mathsf{e}^\omega$. We prove $z_1 \preceq_* z_2$ and $z_2 \preceq_* z_1$; this suffices, because $\preceq_*$ is antisymmetric. To this end, consider the prefix z_1 of $z_1\mathsf{e}^\omega$.

Since $z_1\mathsf{e}^\omega \sqsubseteq z_2\mathsf{e}^\omega$ there is $y \in \mathsf{S}(\tau)$ with $z_1 = \mathsf{it}(z_1) \preceq_* y$. It is easy to see that either $y = z_2$, or $y = \mathsf{e}$, since it maps every prefix of $z_2\mathsf{e}^\omega$ to either one of those. If $y = z_2$, we are done, as $z_1 \preceq_* z_2$. If $y = \mathsf{e}$, then $z_1 \preceq_* \mathsf{e}$, and by antisymmetry of $\preceq_*$, since $\mathsf{e} \preceq_* z_1$ by Definition 6.8, we get $z_1 = \mathsf{e}$ and so we get $z_1 = \mathsf{e} \preceq_* z_2$ again by Definition 6.8. From what we have just proved and $z_2\mathsf{e}^\omega \sqsubseteq z_1\mathsf{e}^\omega$ we then get $z_2 \preceq_* z_1$, hence $z_2 = z_1$ by antisymmetry.

4. Suppose $f : M \to N$ is a left continuous monoid homomorphism, $\sigma, \tau \in M^\omega$, and $\sigma \sqsubseteq \tau$. We need to prove $f^\omega(\sigma) \sqsubseteq f^\omega(\tau)$. To this end, let $x \in \mathsf{S}(f^\omega(\sigma))$. Then $x = \sup_{i\in\mathbb{N}}(\mathsf{it}(f^\star(u_i)))$ for some increasing chain $(u_i)_{i\in\mathbb{N}}$ of prefixes of σ. Since $\sigma \sqsubseteq \tau$, there exists $\sup_{i\in\mathbb{N}}(\mathsf{it}(v_i)) \in \mathsf{S}(\tau)$, with $(v_i)_{i\in\mathbb{N}}$ an increasing sequence of prefixes of τ, and $\sup_{i\in\mathbb{N}}(\mathsf{it}(u_i)) \preceq_* \sup_{i\in\mathbb{N}}(\mathsf{it}(v_i))$. Now, we obtain:

$$\begin{aligned} x &= \sup_{i\in\mathbb{N}}(\mathsf{it}(f^\star(u_i))) \\ &= \sup_{i\in\mathbb{N}}(f(\mathsf{it}(u_i))) \\ &= f(\sup_{i\in\mathbb{N}}(\mathsf{it}(u_i))) \\ &\preceq_* f(\sup_{i\in\mathbb{N}}(\mathsf{it}(v_i))) \\ &= \sup_{i\in\mathbb{N}}(\mathsf{it}(f^\star(v_i))) \end{aligned}$$

The second step holds since f is a monoid homomorphism, the third since f is continuous.

□

EXAMPLE 6.12 : The ω-monoid $C_\infty(A^\star)$ is isomorphic to the (free) ω-monoid $\langle A^\star, A^\infty\rangle$. In fact, the first three ω-monoids in Example 6.3 arise as completions of their underlying monoid, which is left continuous with respect to the natural order $\preceq_*$. For the fourth ($\wp_\omega(X)$) this is the case if X is countable.

Finally, for an ω-monoid $\langle M, X\rangle$, consider the map $\iota^X_M : M \to X$ defined by $\iota^X_M(x) = \mathsf{p}(x\mathsf{e}^\omega)$, where e^ω is the infinite sequence of e's. Thanks to its definition, this map is well-behaved with respect to finite and mixed product, that is, $\iota^X_M(x * y) = x *^{\mathsf{m}} \iota^X_M(y)$, and it is stable under homomorphisms of ω-monoids, that is, if $\langle f, g\rangle : \langle M, X\rangle \to \langle N, Y\rangle$ is an ω-monoid homomorphism, the following diagram commutes:

$$\begin{array}{ccc} M & \xrightarrow{f} & N \\ {\scriptstyle \iota^X_M}\downarrow & & \downarrow{\scriptstyle \iota^Y_N} \\ X & \xrightarrow{g} & Y \end{array}$$

Intuitively, ι^X_M provides a way to embed M into X, however, in general, it is not injective (it depends on the infinite product which can even be a constant function). However, if M is left continuous and the ω-monoid is the completion $C_\infty(M) = \langle M, M_\infty\rangle$, then the map, denoted for short ι_M, becomes injective,

thanks to Lemma 6.11 (3). Therefore, in the sequel of the chapter, we identify M with its image in M_∞, leaving the inclusion ι_M implicit.

We conclude this subsection with the definition of a technical property of ω-monoids, stated for the completion in particular, which will be used to guide the extension of big-step semantics presented in Section 6.3 (cf. Definition 6.35). We note that the use of this property is quite subtle, and most of the extension can be understood without it.

Let $\langle M, *, \mathsf{e}\rangle$ be a (left continuous) monoid. Given $G \subseteq M$, and M_G the submonoid of M generated by G, a sequence $\sigma = (x_i)_{i\in\mathbb{N}}$ in G is *trivial* if it is eventually always the unit, that is, $x_i = \mathsf{e}$ for all $i \geq k$ for some $k \in \mathbb{N}$. Moreover, for $z \in M_\infty$, we define the set $\mathsf{F}(z)$ of *factors* of z in M_G as follows:

$$\mathsf{F}(z) = \{x \in M_G \mid z = y * x *^{\mathsf{m}} \mathsf{p}(\sigma) \text{ for some } y \in M_G \text{ and } \sigma \in M_G^\omega\}$$

We can now define the properties we need:

DEFINITION 6.13 : Let $\langle M, *, \mathsf{e}\rangle$ be a (left continuous) monoid and $G \subseteq M$.

1. We say G has *unique limits* in M_∞ if each non-trivial sequence $\sigma = (x_i)_{i\in\mathbb{N}}$ in G has a unique limit product.
2. We say an element $z \in M_\infty$ is *finitely generated by* G if $\mathsf{F}(z)$ is non-empty and finite.

We report below some properties useful in the following:

LEMMA 6.14 : Let $\sigma = (x_i)_{i\in\mathbb{N}} \in M_G^\omega$ a sequence such that $z = \mathsf{p}(\sigma)$ is finitely generated by G, and, for all $n \in \mathbb{N}$, $u_n = x_0 \dots x_n$. The following hold:

1. there exists $n \in \mathbb{N}$ such that, for all $k \geq n$, $z = \mathsf{p}(u_k \mathsf{e}^\omega)$,
2. for all $n \in \mathbb{N}$, $\sigma = u_n \sigma_{n+1}$ and $\mathsf{p}(\sigma_{n+1})$ is finitely generated.

Proof:

1. Since $z = \mathsf{p}(\sigma)$ is finitely generated, there is $n \in \mathbb{N}$ such that, for all $k \geq n$, $\mathsf{it}(u_n) = \mathsf{it}(u_k)$, because otherwise we would have infinitely many elements in $\mathsf{F}(z)$, that is absurd. Then, $\sigma \equiv u_k \mathsf{e}^\omega$, for all $k \geq n$ trivially holds from what we just observed, hence we have the thesis.
2. Let $\sigma = u_n \sigma_{n+1}$ and note that $\mathsf{F}(\mathsf{p}(\sigma_{n+1}))$ is not empty since the head of σ_{n+1} belongs to it. Then, consider $x \in \mathsf{F}(\mathsf{p}(\sigma_{n+1}))$, by definition of factor we have $\mathsf{p}(\sigma_{n+1}) = x' * x *^{\mathsf{m}} \mathsf{p}(\sigma')$, hence $(\mathsf{it}(u_n) * x') * x *^{\mathsf{m}} \mathsf{p}(\sigma') = \mathsf{p}(\sigma)$, that is, $x \in \mathsf{F}(z)$. This proves that $\mathsf{F}(\mathsf{p}(\sigma_{n+1})) \subseteq \mathsf{F}(z)$, hence $\mathsf{F}(\mathsf{p}(\sigma_{n+1}))$ is finite, and this proves the thesis.

□

6.2.3 Digression: completion from a categorical perspective

In this subsection, we analyse in categorical terms the completion presented in the previous subsection. We will assume basic concepts from category theory, referring, *e.g.*, to the book by Mac Lane (1978).

In the following, we denote by Mon the category of monoids and their homomorphisms, and by ω-Mon the category of ω-monoids and their homomorphisms. There is an obvious forgetful functor $U_{\omega\text{-}\mathsf{Mon}} : \omega\text{-}\mathsf{Mon} \to \mathsf{Mon}$ which forgets the second component of ω-monoids and ω-monoid homomorphisms. First of all, we show that $U_{\omega\text{-}\mathsf{Mon}}$ admits a left adjoint, that is, we define a free construction of an ω-monoid starting from a monoid.

Recall that, given a monoid $\langle M, *, \mathsf{e}\rangle$, $\bowtie$ denotes the relation on M^ω defined as follows: $\sigma \bowtie \tau$ iff there are a decomposition $(u_i)_{i\in\mathbb{N}}$ of σ and a decomposition $(v_i)_{i\in\mathbb{N}}$ of τ such that, for all $i \in \mathbb{N}$, $\mathsf{it}(u_i) = \mathsf{it}(v_i)$. Such relation is trivially reflexive and symmetric, and we denote by $\bowtie^\star$ its transitive closure. The following lemma shows some important properties of $\bowtie^\star$.

LEMMA 6.15 : Let $\langle M, *, \mathsf{e}\rangle$ be a monoid, then, for all $\sigma, \tau \in M^\omega$, the following hold:

1. for all $x \in M$, if $\sigma \bowtie^\star \tau$, then $x\sigma \bowtie^\star x\tau$
2. for all $u \in M^+$, $u\sigma \bowtie^\star \mathsf{it}(u){:}\sigma$
3. if $f : M \to N$ is a monoid homomorphism and $\sigma \bowtie^\star \tau$, then $f^\omega(\sigma) \bowtie^\star f^\omega(\tau)$.

Proof:

1. We prove the thesis for $\bowtie$, then it extends to $\bowtie^\star$ by a straightforward induction. By hypothesis, there are a decomposition $(u_i)_{i\in\mathbb{N}}$ of σ and a decomposition $(v_i)_{i\in\mathbb{N}}$ of τ such that, for all $i \in \mathbb{N}$, $\mathsf{it}(u_i) = \mathsf{it}(v_i)$. Define a decomposition $(u'_i)_{i\in\mathbb{N}}$ of $x\sigma$ and a decomposition $(v'_i)_{i\in\mathbb{N}}$ of $x\tau$, where $u'_0 = v'_0 = x$, $u'_{i+1} = u_i$ and $v'_{i+1} = v_i$. Then, for all $i \in \mathbb{N}$, $\mathsf{it}(u'_i) = \mathsf{it}(v'_i)$ as required.
2. We prove the thesis for $\bowtie$, then it extends to $\bowtie^\star$ by a straightforward induction. Suppose that $\sigma = (x_i)_{i\in\mathbb{N}}$, then it is enough to consider the decomposition $(u_i)_{i\in\mathbb{N}}$ of $u\sigma$ and $(v_i)_{i\in\mathbb{N}}$ of $\mathsf{it}(u){:}\sigma$, where $u_0 = u$, $v_0 = \mathsf{it}(u)$ and $u_{i+1} = v_{i+1} = x_i$.
3. We prove the thesis for $\bowtie$, then it extends to $\bowtie^\star$ by a straightforward induction. By hypothesis, there are a decomposition $(u_i)_{i\in\mathbb{N}}$ of σ and a decomposition $(v_i)_{i\in\mathbb{N}}$ of τ such that, for all $i \in \mathbb{N}$, $\mathsf{it}(u_i) = \mathsf{it}(v_i)$. Hence, $(f^\star(u_i))_{i\in\mathbb{N}}$ is a decomposition of $f^\omega(\sigma)$ and $(f^\star(v_i))_{i\in\mathbb{N}}$ is a decomposition of $f^\omega(\tau)$. Since f is a monoid homomorphism, we get that, for all $i \in \mathbb{N}$, $\mathsf{it}(f^\star(u_i)) = f(\mathsf{it}(u_i))$ and $\mathsf{it}(f^\star(v_i)) = f(\mathsf{it}(v_i))$, hence $\mathsf{it}(f^\star(u_i)) = \mathsf{it}(f^\star(v_i))$, and this implies $f^\omega(\sigma) \bowtie f^\omega(\tau)$.

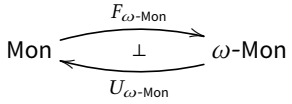
□

Items 1 and 2 of Lemma 6.15 give us well-definedness of the following construction.

DEFINITION 6.16 : Let $\langle M, *, \mathsf{e}\rangle$ be a monoid. We define $F_{\omega\text{-}\mathsf{Mon}}(M)$ to be the ω-monoid $\langle M, M^\omega/{\bowtie^\star}\rangle$ with

- mixed product given by $\langle x, [\sigma]_{\bowtie^\star}\rangle \mapsto [x\sigma]_{\bowtie^\star}$, and
- infinite product given by $\sigma \mapsto [\sigma]_{\bowtie^\star}$.

PROPOSITION 6.17 : The construction in Definition 6.16 extends to a functor $F_{\omega\text{-}\mathsf{Mon}} : \mathsf{Mon} \to \omega\text{-}\mathsf{Mon}$, which is left adjoint to the forgetful functor $U_{\omega\text{-}\mathsf{Mon}}$.



Proof: It is enough to show that, for each monoid M, ω-monoid $\langle N, X\rangle$, and monoid homomorphism $f : M \to N$, there is a unique function $g : M^\omega/{\bowtie^\star} \to X$ such that $\langle f, g\rangle$ is an ω-monoid homomorphism from $F_{\omega\text{-}\mathsf{Mon}}(M)$ to $\langle N, X\rangle$. First of all, note that such a function has to satisfy $g([\sigma]_{\bowtie^\star}) = \mathsf{p}_{N,X}(f^\omega(\sigma))$ for all $\sigma \in M^\omega$, by definition of ω-monoid homomorphism, and this shows uniqueness. It remains to prove that $g = \mathsf{p}_{N,X} \cdot f^\omega$ is well-defined on the quotient $M^\omega/{\bowtie^\star}$, and that it is compatible with the mixed product. Towards a proof of the former point, we have to prove that, for all $\sigma, \tau \in M^\omega$, if $\sigma \bowtie^\star \tau$, then $\mathsf{p}_{N,X}(f^\omega(\sigma)) = \mathsf{p}_{N,X}(f^\omega(\tau))$. By Lemma 6.15 (3), we get $f^\omega(\sigma) \bowtie^\star f^\omega(\tau)$, and by the infinite associative law we get $\mathsf{p}_{N,X}(f^\omega(\sigma)) = \mathsf{p}_{N,X}(f^\omega(\tau))$. Compatibility with the mixed product follows by

$$\mathsf{p}_{N,X}(f^\omega(x\sigma)) = \mathsf{p}_{N,X}(f(x)f^\omega(\sigma)) = f(x) *^{\mathsf{m}} \mathsf{p}_{N,X}(f^\omega(\sigma)).$$

□

Note that, since $\langle A^\star, \cdot, \varepsilon\rangle$ is the free monoid over the set A, $F_{\omega\text{-}\mathsf{Mon}}(A^\star)$ is the free ω-monoid over the set A, just by composing the adjunctions. Therefore, to prove Proposition 6.4, we just have to prove that $F_{\omega\text{-}\mathsf{Mon}}(A^\star)$ is isomorphic (in ω-Mon) to $\langle A^\star, A^\infty\rangle$, as done below.

Proof (Proposition 6.4)*:* Recall that the infinite product p of the ω-monoid $\langle A^\star, A^\infty\rangle$ acts by flatterning, that is, maps an infinite sequence $(u_i)_{i\in\mathbb{N}} \in (A^\star)^\omega$ to $u_0u_1u_2\ldots \in A^\infty$. In other words, it can be defined corecursively by the following equations: $\mathsf{p}(\varepsilon^\omega) = \varepsilon$ and $\mathsf{p}(\varepsilon^n \cdot ((xu){:}\sigma)) = x \cdot \mathsf{p}(u{:}\sigma)$.

By Proposition 6.17, we know that there is a unique ω-monoid homomorphism $\langle f, g\rangle : F_{\omega\text{-}\mathsf{Mon}}(A^\star) \to \langle A^\star, A^\infty\rangle$ such that $f = \mathsf{id}_{A^\star}$, and we know that g must act as p (on equivalence classes), because $\mathsf{id}^\omega_{A^\star} = \mathsf{id}_{(A^\star)^\omega}$. We have to construct an inverse of g. We can corecursively define a function $e : A^\infty \to (A^\star)^\omega$ by the following equations: $e(\varepsilon) = \varepsilon^\omega$ and $e(x\alpha) = x{:}e(\alpha)$. It is easy to check that $\langle \mathsf{id}_{A^\star}, e\rangle$ is an ω-monoid homomorphism from $\langle A^\star, A^\infty\rangle$ to $F_{\omega\text{-}\mathsf{Mon}}(A^\star)$. Furthermore, it can be checked, by coinduction, that $\sigma \bowtie e(\mathsf{p}(\sigma))$, for all $\sigma \in (A^\star)^\omega$, because, if $\sigma = u{:}\sigma'$, we have $e(\mathsf{p}(\sigma)) = u \cdot e(\mathsf{p}(\sigma'))$, and $\alpha = \mathsf{p}(e(\alpha))$, for all $\alpha \in A^\infty$, because, if $\alpha = \varepsilon$, $\mathsf{p}(e(\varepsilon)) = \varepsilon$ and, if $\alpha = x{:}\alpha'$, $\mathsf{p}(e(\alpha)) = x{:}\mathsf{p}(e(\alpha'))$. Therefore, we have the thesis. □

We now present the completion of the previous subsection in categorical terms, relating it to the free construction. Let us denote by LMon the category

of left continuous monoids and continuous homomorphisms, which is trivially a subcategory of Mon, as shown by the inclusion functor $I_{\mathsf{LMon}} : \mathsf{LMon} \longrightarrow \mathsf{Mon}$. We define left continuous ω-monoids as follows:

DEFINITION 6.18 (Left continuous ω-monoid): An ω-monoid $\langle M, X\rangle$ is left continuous if

- M is left continuous and
- the infinite product p satisfies the *continuous infinite associative law*: for all $\sigma, \tau \in M^\omega$, if $\sigma \equiv \tau$, then $\mathsf{p}(\sigma) = \mathsf{p}(\tau)$.

An ω-monoid homomorphism $\langle f, g\rangle$ between left continuous ω-monoids is continuous if so is f.

We denote by ω-LMon the category of left continuous ω-monoids and continuous homomorphisms, which is trivially a subcategory of ω-Mon, as witnessed by the inclusion functor $I_{\omega\text{-}\mathsf{LMon}} : \omega\text{-}\mathsf{LMon} \longrightarrow \omega\text{-}\mathsf{Mon}$, which is well-defined by Lemma 6.11 (2). Furthermore, there is an obvious forgetful functor $U_{\omega\text{-}\mathsf{LMon}} : \omega\text{-}\mathsf{LMon} \longrightarrow \mathsf{LMon}$, which forgets the second component of left continuous ω-monoids and continuous homomorphisms. It is easy to check that the following diagram commutes:

$$\begin{array}{ccc} \mathsf{Mon} & \xleftarrow{U_{\omega\text{-}\mathsf{Mon}}} & \omega\text{-}\mathsf{Mon} \\ {\scriptstyle I_{\mathsf{LMon}}}\uparrow & & \uparrow{\scriptstyle I_{\omega\text{-}\mathsf{LMon}}} \\ \mathsf{LMon} & \xleftarrow[U_{\omega\text{-}\mathsf{LMon}}]{} & \omega\text{-}\mathsf{LMon} \end{array}$$

Note that, given a left-continuous monoid M, the ω-monoid $\langle M, M_\infty\rangle$, constructed in Definition 6.10, is actually a left continuous ω-monoid. Hence, we get the following key proposition.

PROPOSITION 6.19 : The construction in Definition 6.10 extends to a functor $F_{\omega\text{-}\mathsf{LMon}} : \mathsf{LMon} \longrightarrow \omega\text{-}\mathsf{LMon}$, which is left adjoint to the forgetful functor $U_{\omega\text{-}\mathsf{LMon}}$.

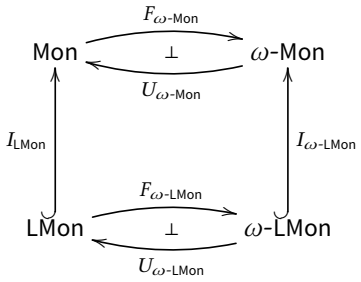
$$\mathsf{LMon} \underset{U_{\omega\text{-}\mathsf{LMon}}}{\overset{F_{\omega\text{-}\mathsf{LMon}}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \omega\text{-}\mathsf{LMon}$$

Proof: It is enough to show that, for each left continuous monoid M, left continuous ω-monoid $\langle N, X\rangle$ and continuous monoid homomorphism $f : M \longrightarrow N$, there is a unique function $g : M^\omega/{\equiv} \longrightarrow X$ such that $\langle f, g\rangle$ is an ω-monoid homomorphism from $F_{\omega\text{-}\mathsf{LMon}}(M) = \langle M, M_\infty\rangle$ to $\langle N, X\rangle$ (it is trivially continuous as so is f).

First of all, note that such a function has to satisfy $g([\sigma]_\equiv) = \mathsf{p}_{N,X}(f^\omega(\sigma))$ for all $\sigma \in M^\omega$, by definition of ω-monoid homomorphism, and this shows uniqueness. It remains to prove that $g = \mathsf{p}_{N,X} \cdot f^\omega$ is well-defined on the quotient $M^\omega/{\equiv}$, and that it is compatible with the mixed product. Towards a proof of the former point, we have to prove that, for all $\sigma, \tau \in M^\omega$, if $\sigma \equiv \tau$, then $\mathsf{p}_{N,X}(f^\omega(\sigma)) = \mathsf{p}_{N,X}(f^\omega(\tau))$. By Lemma 6.11 (4), we get $f^\omega(\sigma) \equiv f^\omega(\tau)$, and by the continuous infinite associative law we get

$\mathsf{p}_{N,X}(f^\omega(\sigma)) = \mathsf{p}_{N,X}(f^\omega(\tau))$. Compatibility with the mixed product follows by $\mathsf{p}_{N,X}(f^\omega(x\sigma)) = \mathsf{p}_{N,X}(f(x)f^\omega(\sigma)) = f(x) *^{\mathsf{m}} \mathsf{p}_{N,X}(f^\omega(\sigma))$. □

COROLLARY 6.20 : We have the following diagram



where the following hold:

- $I_{\mathsf{LMon}} \cdot U_{\omega\text{-LMon}} = U_{\omega\text{-Mon}} \cdot I_{\omega\text{-LMon}}$,
- $U_{\omega\text{-Mon}} \cdot F_{\omega\text{-Mon}} = \mathrm{Id}_{\mathsf{Mon}}$ and $U_{\mathsf{LMon}} \cdot F_{\omega\text{-LMon}} = \mathrm{Id}_{\mathsf{LMon}}$,
- there is a surjective natural transformation $(\phi_M) : F_{\omega\text{-Mon}}(I_{\mathsf{LMon}}(M)) \to I_{\omega\text{-LMon}}(F_{\omega\text{-LMon}}(M))$.

Proof: The first two items are trivial. The third one follows by defining ϕ_M as the component at $I_{\mathsf{LMon}}(M)$ of the counit of the adjunction $F_{\omega\text{-Mon}} \dashv U_{\omega\text{-Mon}}$, because we have $I_{\mathsf{LMon}}(M) = U_{\omega\text{-Mon}}(I_{\omega\text{-LMon}}(F_{\omega\text{-LMon}}(M)))$. □

Finally, we note that the completion C_∞ from left continuous monoids to ω-monoids is actually a functor, notably the composite of $F_{\omega\text{-LMon}}$ followed by $I_{\omega\text{-LMon}}$.

6.3 Extending big-step semantics with observations

In this section, we start defining big-step semantics with observations (Definition 6.21) and, following the approach of Section 5.1, what are computations in such a big-step semantics. Then, we will formally define the construction extending a big-step semantics with observations from finite to infinite computations.

6.3.1 Definition

We start by providing a general formal definition of big-step semantics with observations, following Definition 5.1.

DEFINITION 6.21 (Big-step semantics with observations): A *big-step semantics with observations* is a tuple $\langle C, R, O, \mathcal{R}\rangle$ where:

- C is a set of *configurations* c,
- R is a set of *results* r,

- O is a left continuous monoid of *(finite) observations* o, with (finite) product $*$ and identity e. We define *judgments* $j = c \Rightarrow \langle r, o \rangle$, meaning that configuration c evaluates to result r producing the observation o. Set $C(j) = c$, $R(j) = r$ and $O(j) = o$.
- $\mathcal{R}$ is a set of *(big-step) rules* ρ of shape

$$\frac{j_1 \quad \dots \quad j_n}{c \Rightarrow \langle r, o_0 * O(j_1) * \dots * o_{n-1} * O(j_n) * o_n \rangle}$$

  also written in *inline format*: $\mathsf{rule}(j_1 \dots j_n,\ o_0 \dots o_n,\ c,\ r)$, where $j_1 \dots j_n$, with $n \geq 0$, is a sequence of premises and $o_0 \dots o_n$ is a sequence of *elementary* observations. Set $C(\rho) = c$, $R(\rho) = r$, $O(\rho) = o_0 * O(j_1) * \dots * o_{n-1} O(j_n) * o_n$, for $i \in 1..n$, $C(\rho, i) = C(j_i)$, $R(\rho, i) = R(j_i)$ and $O(\rho, i) = O(j_i)$ and, for all $i \in 0..n$, $E(\rho, i) = o_i$.

We will use the inline format, more concise and manageable, for the development of the meta-theory, e.g., in constructions.

As discussed for standard big-step semantics in Section 5.1, big-step rules defined above are very much like inference rules in Definition 2.1. However, they carry a richer structure. Notably, premises are a sequence, rather than a set, hence, they are ordered and can be repeated, and the sequence of elementary observations is made explicit. Again, such additional structure does not affect derivability using these rules, but it is essential to develop the meta-theory in this chapter, notably, to define computations and the construction. Indeed, as premises are a sequence, we know in which order configurations in the premises should be evaluated and, as the sequence of elementary observations is explicit, we know when and which observation should be produced.

Therefore, given a big-step semantics with observations $\langle C, R, O, \mathcal{R} \rangle$, slightly abusing the notation, we denote by $\mathcal{R}$ the inference system obtained by forgetting such additional structure, and define, as usual, the *semantic relation* as the inductive interpretation of $\mathcal{R}$. As customary, in the following we will write $\mathcal{R} \vdash_\mu c \Rightarrow \langle r, o \rangle$ when the judgment $c \Rightarrow \langle r, o \rangle$ is inductively derivable in $\mathcal{R}$, namely, it has a finite derivation.

Again, as for standard big-step semantics, since in practice the (infinite) set of rules $\mathcal{R}$ is described by a finite set of meta-rules, each one with a finite number of premises, the number of premises of rules is not only finite but *bounded*. We model this feature (relevant in the following) by an explicit assumption, essentially equal to Assumption 5.1:

ASSUMPTION 6.1 (Bounded premises (BP)): For a big-step semantics with observations $\langle C, R, O, \mathcal{R} \rangle$, there exists $b \in \mathbb{N}$ such that

$$\text{for each } \rho = \mathsf{rule}(j_1 \dots j_n,\ o_0 \dots o_n,\ c,\ r),\ n \leq b.$$

EXAMPLE 6.22 : The big-step semantics of λ-calculus with output in Figure 6.1 is an instance of Definition 6.21 where

- configurations are expressions e,
- results are values v,

- observations are finite sequences of values $v_1 \dots v_n$, with finite product and identity given by concatenation and the empty sequence, respectively,
- rules are those of Figure 6.1 where we assume a left-to-right order on premises and where we have omitted elementary observations equal to the empty list. In the inline format, rules are the following:

  (VAL) $\mathsf{rule}(\varepsilon,\ \varepsilon,\ v,\ v)$

  (APP) $\mathsf{rule}(e_1 \Rightarrow \langle \lambda x.e, o_1\rangle\ e_2 \Rightarrow \langle v_2, o_2\rangle\ e[v_2/x] \Rightarrow \langle v, o\rangle,\ \varepsilon\varepsilon\varepsilon\varepsilon,\ e_1\ e_2,\ v)$

  (OUT) $\mathsf{rule}(e \Rightarrow \langle v, o_1\rangle,\ \varepsilon v,\ \mathtt{out}\ e,\ v)$

EXAMPLE 6.23 : As discussed on page 73, rule (APP) formalises for an application $e_1\ e_2$ left-to-right evaluation with early error detection. Right-to-left evaluation can be expressed by just swapping the first two premises, that is:

(APP-RIGHT) $\mathsf{rule}(e_2 \Rightarrow \langle v_2, o_2\rangle\ e_1 \Rightarrow \langle \lambda x.e, o_1\rangle\ e[v_2/x] \Rightarrow \langle v, o\rangle,\ \varepsilon\varepsilon\varepsilon\varepsilon,\ e_1\ e_2,\ v)$

Left-to-right evaluation with late error detection can be expressed as follows:

(APP-LATE) $\mathsf{rule}(e_1 \Rightarrow \langle v_1, o_1\rangle\ e_2 \Rightarrow \langle v_2, o_2\rangle\ v_1 \Rightarrow \langle \lambda x.e, o_3\rangle\ e[v_2/x] \Rightarrow \langle v, o\rangle,\ \varepsilon\varepsilon\varepsilon\varepsilon\varepsilon,\ e_1\ e_2,\ v)$

6.3.2 Computations

As discussed in Chapter 5, the dynamics of the evaluation process is implicit in big-step rules. This is also the case when we extend the definition by observations, hence, in this section, following the approach of Section 5.2, we make it explicit by defining *computations* for a big-step semantics with observations.

We have defined computations for a standard big-step semantics (cf. Section 5.2) by introducing a transition relation between partial evaluations that formally models the evaluation algorithm implicitly associated with a big-step semantics. Here, we extend this approach to big-step semantics with observations. Notably, the main difference is that in this setting the transition relation will be *labelled* by an observation, which is the one emitted by this evaluation step. In this way, a computation, that is, a (possibly infinite) sequence of steps, will be naturally associated with a (possibly infinite) sequence of observations that can be interpreted in the completion of the monoid of finite observations, as we will formally explain.

First of all, note that a big-step semantics with observations $\langle C, R, O, \mathcal{R}\rangle$ trivially determines a big-step semantics as defined in Definition 5.1, by forgetting elementary observations. More precisely, the resulting big-step semantics is $\langle C, R \times O, \overline{\mathcal{R}}\rangle$, where a rule $\mathsf{rule}(j_1 \dots j_n,\ c,\ \langle r, o\rangle)$ belongs to $\overline{\mathcal{R}}$ iff $\mathsf{rule}(j_1 \dots j_n,\ o_0 \dots o_n,\ c,\ r)$ belongs to $\mathcal{R}$ and $o = o_0 * O(j_1) * \dots * o_{n-1} * O(j_n) * o_n$, for some $o_0, \dots, o_n \in O$. Hence, all definitions and results developed for standard big-step semantics can be applied to big-step semantics with observations as well. In particular, we can use Definition 5.2 to extend big-step rules with ? and so construct partial evaluation trees (cf. Definition 5.4) for big-step semantics with observations, thus modelling incomplete evaluations, and, finally, use them to define the transition relation.

$$\text{(LTR-1)}\quad \mathsf{ax}_?(c)\,\frac{}{c \Rightarrow ?} \xrightarrow{o}_{\mathcal{R}} (\rho)\,\frac{}{c \Rightarrow \langle r, o\rangle} \quad \rho = \mathsf{rule}(\varepsilon,\ o,\ c,\ r)$$

$$\text{(LTR-2)}\quad \mathsf{ax}_?(c)\,\frac{}{c \Rightarrow ?} \xrightarrow{o}_{\mathcal{R}} (\mathsf{pev}_?(\rho, 1, ?))\,\frac{c' \Rightarrow ?}{c \Rightarrow ?} \quad \begin{array}{l} C(\rho) = c,\ C(\rho, 1) = c' \\ E(\rho, 0) = o \end{array}$$

$$\text{(LTR-3)}\quad (\mathsf{pev}_?(\rho, i, r))\,\frac{\tau_1\ \ldots\ \tau_i}{c \Rightarrow ?} \xrightarrow{o}_{\mathcal{R}} (\rho')\,\frac{\tau_1\ \ldots\ \tau_i}{c \Rightarrow \langle r', o'\rangle} \quad \begin{array}{l} \rho' \sim_i \rho,\ \#\rho' = i \\ R(\rho', i) = r,\ E(\rho', i) = o \\ R(\rho') = r',\ O(\rho') = o' \end{array}$$

$$\text{(LTR-4)}\quad (\mathsf{pev}_?(\rho, i, r))\,\frac{\tau_1\ \ldots\ \tau_i}{c \Rightarrow ?} \xrightarrow{o}_{\mathcal{R}} (\mathsf{pev}_?(\rho', i+1, ?))\,\frac{\tau_1\ \ldots\ \tau_i\ c' \Rightarrow ?}{c \Rightarrow ?} \quad \begin{array}{l} \rho' \sim_i \rho \\ R(\rho', i) = r,\ E(\rho', i) = o \\ C(\rho', i+1) = c' \end{array}$$

$$\text{(LTR-5)}\quad (\mathsf{pev}_?(\rho, i, ?))\,\frac{\tau_1\ \ldots\ \tau_{i-1}\ \tau_i}{c \Rightarrow ?} \xrightarrow{o}_{\mathcal{R}} (\mathsf{pev}_?(\rho, i, r))\,\frac{\tau_1\ \ldots\ \tau_{i-1}\ \tau_i'}{c \Rightarrow ?} \quad \tau_i \xrightarrow{o}_{\mathcal{R}} \tau_i'$$

FIGURE 6.6 Labelled transition relation between partial evaluation trees.

Figure 6.6 contains rules defining the labelled transition relation associated with a big-step semantics with observations. As without observations, this transition relation is defined on partial evaluation trees *annotated* by rules: each node is associated with the rule in $\mathcal{R}_?$ used to derive it from its children. However, in this context the annotation is not redundant, because rules in $\mathcal{R}$ carry more information (the elementary observations), and annotations will be important also for the subsequent formal development.

Similarly to the transition relation for standard big-step semantics (cf. Figure 5.3), it relies on an equivalence relation on rules modelling equality up-to an index, which, in this context, has to take into account elementary observations as well:

DEFINITION 6.24 : Given rules in $\mathcal{R}$

$$\rho = \mathsf{rule}(j_1 \ldots j_n,\ o_0 \ldots o_n,\ c,\ r) \text{ and } \rho' = \mathsf{rule}(j_1' \ldots j_m',\ o_0' \ldots o_m',\ c',\ r'),$$

for any index $i \in 1..\min(n, m)$, define $\rho \sim_i \rho'$ if and only if

- $c = c'$,
- for all $k < i$, $j_k = j_k'$ and $o_k = o_k'$, and
- $C(j_i) = C(j_i')$.

Intuitively, this means that rules ρ and ρ' model the same computation until the i-th configuration included.

As without observations, each transition step makes "less incomplete" the partial evaluation tree, but, differently, it produces an (elementary) observation according to the selected rule. Notably, transition rules apply only to nodes labelled by incomplete judgements ($c \Rightarrow ?$), whereas subtrees whose root is a complete judgement ($c \Rightarrow \langle r, o\rangle$) cannot move. In detail:

- If the last applied rule is $\mathsf{ax}_?(c)$, then we have to find a rule ρ with c in the conclusion and, if it has no premises we just return $\langle R(\rho), O(\rho)\rangle$ as

$$\frac{}{id\,(\mathtt{out}\,0) \Rightarrow ?} \xrightarrow{\varepsilon}_{\mathcal{R}} \frac{id \Rightarrow ?}{id\,(\mathtt{out}\,0) \Rightarrow ?} \xrightarrow{\varepsilon}_{\mathcal{R}} \frac{id \Rightarrow \langle id, \varepsilon\rangle}{id\,(\mathtt{out}\,0) \Rightarrow ?}$$

$$\xrightarrow{\varepsilon}_{\mathcal{R}} \frac{id \Rightarrow \langle id, \varepsilon\rangle \quad \mathtt{out}\,0 \Rightarrow ?}{id\,(\mathtt{out}\,0) \Rightarrow ?} \xrightarrow{\varepsilon}_{\mathcal{R}} \frac{id \Rightarrow \langle id, \varepsilon\rangle \quad \dfrac{0 \Rightarrow ?}{\mathtt{out}\,0 \Rightarrow ?}}{id\,(\mathtt{out}\,0) \Rightarrow ?}$$

$$\xrightarrow{\varepsilon}_{\mathcal{R}} \frac{id \Rightarrow \langle id, \varepsilon\rangle \quad \dfrac{0 \Rightarrow \langle 0, \varepsilon\rangle}{\mathtt{out}\,0 \Rightarrow ?}}{id\,(\mathtt{out}\,0) \Rightarrow ?} \xrightarrow{0}_{\mathcal{R}} \frac{id \Rightarrow \langle id, \varepsilon\rangle \quad \dfrac{0 \Rightarrow \langle 0, \varepsilon\rangle}{\mathtt{out}\,0 \Rightarrow \langle 0, 0\rangle}}{id\,(\mathtt{out}\,0) \Rightarrow ?}$$

$$\xrightarrow{\varepsilon}_{\mathcal{R}} \frac{id \Rightarrow \langle id, \varepsilon\rangle \quad \dfrac{0 \Rightarrow \langle 0, \varepsilon\rangle}{\mathtt{out}\,0 \Rightarrow \langle 0, 0\rangle} \quad 0 \Rightarrow ?}{id\,(\mathtt{out}\,0) \Rightarrow ?}$$

$$\xrightarrow{\varepsilon}_{\mathcal{R}} \frac{id \Rightarrow \langle id, \varepsilon\rangle \quad \dfrac{0 \Rightarrow \langle 0, \varepsilon\rangle}{\mathtt{out}\,0 \Rightarrow \langle 0, 0\rangle} \quad 0 \Rightarrow \langle 0, \varepsilon\rangle}{id\,(\mathtt{out}\,0) \Rightarrow ?}$$

$$\xrightarrow{\varepsilon}_{\mathcal{R}} \frac{id \Rightarrow \langle id, \varepsilon\rangle \quad \dfrac{0 \Rightarrow \langle 0, \varepsilon\rangle}{\mathtt{out}\,0 \Rightarrow \langle 0, 0\rangle} \quad 0 \Rightarrow \langle 0, \varepsilon\rangle}{id\,(\mathtt{out}\,0) \Rightarrow \langle 0, 0\rangle}$$

FIGURE 6.7 The evaluation of $id\,(\mathtt{out}\,0)$, with $id = \lambda x.x$, using $\xrightarrow{o}_{\mathcal{R}}$ for rules in Figure 6.1.

result and producing as observation $O(\rho)$[6], otherwise we start evaluating the first premise of such rule, producing the first elementary observation $E(\rho, 0)$.

- If the last applied rule is $\mathsf{pev}_?(\rho, i, r)$, then all subtrees are complete, hence, to continue the evaluation, we have to find another rule ρ', having, for each $k \in 1..i$, as k-th premise the root of τ_k. Then there are two possibilities: if there is an $i+1$-th premise, then we start evaluating it, otherwise we return $\langle R(\rho'), O(\rho')\rangle$ as result, and, in both cases, we produce the i-th elementary observation $E(\rho', i)$.
- If the last applied rule is a propagation rule $\mathsf{pev}_?(\rho, i, ?)$, then we simply propagate the step, and the produced observation, made by τ_i (the last subtree).

As usual, we extend $\longrightarrow_{\mathcal{R}}$ to sequences of observations, writing $\tau \xrightarrow{u}{}^{\star}_{\mathcal{R}} \tau'$, with $u \in O^{\star}$, when we can reach τ' from τ producing the sequence of observations u.

In Figure 6.7 we report an example of evaluation of a term according to rules in Figure 6.1, using partial evaluation trees and $\longrightarrow_{\mathcal{R}}$.

As said above and as happens without observations, the definition of $\xrightarrow{o}_{\mathcal{R}}$ given in Figure 6.6 nicely models as a transition system an interpreter driven by the big-step rules, specifying an algorithm of incremental evaluation, which

6 Note that, since ρ has no premises, $O(\rho)$ is an elementary observation, that is, $E(\rho, 0) = O(\rho)$.

$$O\left(\left(\mathsf{ax}_?(c)\right)\frac{}{c \Rightarrow ?}\right) = \mathsf{e} \qquad O\left(\left(\rho\right)\frac{\tau_1 \ \ldots \ \tau_n}{c \Rightarrow \langle r, o\rangle}\right) = o$$

$$O\left(\left(\mathsf{pev}_?(\rho, i, r_?)\right)\frac{\tau_1 \ \ldots \ \tau_i}{c \Rightarrow ?}\right) = \prod_{k \in 1..i} E(\rho, k-1) * O(\tau_k)$$

FIGURE 6.8 Inductive definition of $O(\tau)$.

at each step produces an observation. We now show that the labelled transition relation $\longrightarrow_{\mathcal{R}}$ agrees with the semantic relation (inductively) defined by $\mathcal{R}$, namely, the semantic relation captures exactly successful terminating computations in $\longrightarrow_{\mathcal{R}}$. To prove this result (Theorem 6.28), we need some preliminary definitions and lemmas.

Figure 6.8 shows equations inductively defining the *partial observation* $O(\tau)$ of a finite (annotated) partial evaluation tree τ. Intuitively, $O(\tau)$ represents the observation produced by the partial evaluation modelled by τ. The next lemma shows that partial observations grow when performing a computation step by $\longrightarrow_{\mathcal{R}}$.

LEMMA 6.25 : If $\tau \xrightarrow{o}_{\mathcal{R}} \tau'$, then $O(\tau') = O(\tau) * o$.

Proof: The proof is by induction on the definition of $\longrightarrow_{\mathcal{R}}$.

Case: (LTR-1) We have $O(\tau) = \mathsf{e}$ and $O(\tau') = o$, hence the thesis holds.

Case: (LTR-2) We have $O(\tau) = \mathsf{e}$ and $O(\tau') = E(\rho, 0) = o$, hence the thesis holds.

Case: (LTR-3) We have $O(\tau) = \prod_{k \in 1..i} E(\rho, k) * O(\tau_k)$ and $O(\tau') = O(\mathsf{r}(\tau')) = (\prod_{k \in 1..i} E(\rho, k-1) * O(\tau_k)) * E(\rho, i)$, because all τ_k are complete. Since $\rho \sim_i \rho'$, we have $E(\rho, k-1) = E(\rho', k-1)$, for all $k \in 1..i$, and, since $E(\rho, i) = o$, we conclude $O(\tau') = O(\tau) * o$.

Case: (LTR-4)
We have $O(\tau) = \prod_{k \in 1..i} E(\rho, k) * O(\tau_k)$ and $O(\tau') = (\prod_{k \in 1..i} E(\rho, k-1) * O(\tau_k)) * E(\rho, i)$. Since $\rho \sim_i \rho'$, we have $E(\rho, k-1) = E(\rho', k-1)$, for all $k \in 1..i$, and, since $E(\rho, i) = o$, we conclude $O(\tau') = O(\tau) * o$.

Case: (LTR-5)
We have $O(\tau) = \prod_{k \in 1..i} E(\rho, k) * O(\tau_k)$ and $O(\tau') = (\prod_{k \in 1..i-1} E(\rho, k-1) * O(\tau_k)) * E(\rho, i-1) * O(\tau_i')$. By induction hypothesis, we get $O(\tau_i') = O(\tau_i) * o$, hence we get the thesis.

□

LEMMA 6.26 : If $\tau \xrightarrow{u}{}^{\star}_{\mathcal{R}} \tau'$, then $O(\tau') = O(\tau) * \mathsf{it}(u)$.

Proof: Straightforward induction on u, using Lemma 6.25. □

LEMMA 6.27 : If τ is a finite partial evaluation tree with $C(\mathsf{r}(\tau)) = c$, then $Rule c \Rightarrow ? \xrightarrow{u}{}^{\star}_{\mathcal{R}} \tau$ for some $u \in O^{\star}$ with $\mathsf{it}(u) = O(\tau)$.

Proof: The proof is by induction on the height of τ. We have three cases.

- If $\tau = \dfrac{}{c \Rightarrow ?}$, the thesis is trivial by taking $u = \varepsilon$.
- If $\tau = {}_{(\mathsf{pev}_?(\rho, i, R^O_?(\mathsf{r}(\tau_i))))}\dfrac{\tau_1 \ \ldots \ \tau_i}{c \Rightarrow ?}$, with $\rho = \mathsf{rule}(j_1 \ldots j_n,\ o_0 \ldots o_n,\ c,\ r)$ and $i \in 1..n$ and $c_k = C(j_k) = C(\mathsf{r}(\tau_k))$, for all $k \in 1..i$, then by induction hypothesis, we get $\dfrac{}{c_k \Rightarrow ?} \xrightarrow{u_k}{}^{\star}_{\mathcal{R}} \tau_k$, for all $k \in 1..i$. For all $k \in 0..i$ let us define

$$\begin{array}{lll} & \tau'_0 = {}_{(\mathsf{ax}_?(c))}\dfrac{}{c \Rightarrow ?} & v_0 = \varepsilon \\ (k \geq 1) & \tau'_k = {}_{(\mathsf{pev}_?(\rho, k, R^O_?(\mathsf{r}(\tau_k))))}\dfrac{\tau_1 \ \ldots \ \tau_k}{c \Rightarrow ?} & v_k = v_{k-1} o_{k-1} u_k \end{array}$$

  By induction on $k \in 0..i$, we can show that $\tau'_0 \xrightarrow{v_k}{}^{\star}_{\mathcal{R}} \tau'_k$, for all $k \in 0..i$. For $k = 0$ the thesis is trivial, as $v_{=}\varepsilon$. For $k \geq 1$, by induction hypothesis, we get $\tau'_0 \xrightarrow{v_{k-1}}{}^{\star}_{\mathcal{R}} \tau'_{k-1}$, then, by applying (LTR-4) and, since $\dfrac{}{c_k \Rightarrow ?} \xrightarrow{u_k}{}^{\star}_{\mathcal{R}} \tau_k$, by iteratively applying (LTR-5), we get $\tau'_{k-1} \xrightarrow{o_{k-1} u_k}{}^{\star}_{\mathcal{R}} \tau'_k$, hence $\tau'_0 \xrightarrow{v_k}{}^{\star}_{\mathcal{R}} \tau'_k$, as needed. THerefore, in particular we have $\tau'_0 \xrightarrow{v_i}{}^{\star}_{\mathcal{R}} \tau'_i = \tau$, hence, to conclude, we have just to note that $\mathsf{it}(v_i) = \prod_{k \in 1..i} o_{k-1} * \mathsf{it}(u_k)$ and, as $\mathsf{it}(u_k) = O(\tau_k)$ for all $k \in 1..i$, we get $\mathsf{it}(v_i) = O(\tau_i)$, as needed.
- If $\tau = {}_{(\rho)}\dfrac{\tau_1 \ \ldots \ \tau_n}{c \Rightarrow \langle r, o \rangle}$, with $\rho = \mathsf{rule}(j_1 \ldots j_n,\ o_0 \ldots o_n,\ c,\ r)$, then by rule (LTR-3) or (LTR-1), we have $\tau' \xrightarrow{o_n}{}_{\mathcal{R}} \tau$ and τ' is incomplete and has the same height of τ, thus, by previous items we have $Rule c \Rightarrow ? \xrightarrow{u}{}^{\star}_{\mathcal{R}} \tau'$, with $\mathsf{it}(u) = O(\tau')$. By Lemma 6.25, we get $O(\tau) = O(\tau') * o_n = \mathsf{it}(u o_n)$, as needed.

□

THEOREM 6.28 : $\mathcal{R} \vdash_{\mu} c \Rightarrow \langle r, o \rangle$ iff $\dfrac{}{c \Rightarrow ?} \xrightarrow{u}{}^{\star}_{\mathcal{R}} \tau$, with $\mathsf{it}(u) = o$ and $\mathsf{r}(\tau) = c \Rightarrow \langle r, o \rangle$.

Proof: The left-to-right implication follows by Lemma 6.27, because if $\mathcal{R} \vdash_{\mu} c \Rightarrow \langle r, o \rangle$, by definition, there is a finite evaluation tree τ with $\mathsf{r}(\tau) = c \Rightarrow \langle r, o \rangle$, hence $O(\tau) = o$.

The right-to-left implication is immediate because $\mathsf{it}(u) = O(\tau) = o$, by Lemma 6.26 and, since $\mathsf{r}(\tau) = c \Rightarrow \langle r, o \rangle$, τ is a correct finite derivation for $c \Rightarrow \langle r, o \rangle$, hence $\mathcal{R} \vdash_{\mu} c \Rightarrow \langle r, o \rangle$ holds. □

6.3.3 Construction

We now turn to the extension of a big-step semantics with observations $\langle C, R, O, \mathcal{R} \rangle$, which, similarly to the construction in Section 5.4, consists in the addition of rules for divergence propagation as well as corules to rule out spurious results, as shown in the example in Section 6.1. Due to the presence of observations, the choice of corules to add is less trivial than the case without observations (cf. Definition 5.26). Indeed, depending on properties of $\mathcal{R}$ and the observation monoid O, we will study two possible choices for corules to be added—see the final construction in Definition 6.35.

First of all, we add a special result ∞ modelling divergence and we consider the ω-monoid $\langle O, O_\infty \rangle$, obtained by completion of O. Note that, since O is left continuous, we have $O \subseteq O_\infty$.[7] Then, the extended judgement has shape $c \Rightarrow \langle r_\infty, o_\infty \rangle$ where $r_\infty \in R_\infty = R + \{\infty\}$ and $o_\infty \in O_\infty$.

We start by defining basic rules for divergence propagation.

DEFINITION 6.29 (Rules for divergence): The set of rules $\mathcal{R}_\infty$ is obtained by adding to $\mathcal{R}$ the following rules:

DIVERGENCE PROPAGATION RULES For each $\rho = \mathsf{rule}(j_1 \dots j_n,\ o_0 \dots o_n,\ c,\ r)$ in $\mathcal{R}$, index $i \in 1..n$ and possibly infinite observation $o_\infty \in O_\infty$, define rule $\mathsf{div}(\rho,\ i,\ o_\infty)$ as

$$\frac{j_1 \quad \dots \quad j_{i-1} \quad C(j_i) \Rightarrow \langle \infty, o_\infty \rangle}{c \Rightarrow \langle \infty, o_0 * O(j_1) * \dots * o_{i-2} * O(j_{i-1}) * o_{i-1} *^{\mathsf{m}} o_\infty \rangle}$$

Intuitively, we consider the possibility that evaluation of $C(j_i)$ diverges, for one of the premises j_i, producing a possibly infinite observation o_∞. In that case, the subsequent premises should be ignored and the configuration c in the conclusion should diverge as well.

As already mentioned, choosing the appropriate set of corules, that is, a set of corules providing the "expected" semantics, is not trivial at all. However, there is a first basic property that any set of corules must satisfy to properly model divergence: the resulting semantics must be *conservative*, that is, do not change the semantics of finite computations.

We now describe a "syntactic" property of corules ensuring the resulting semantics to be conservative.

DEFINITION 6.30 : A set $\mathcal{C}^{\mathcal{R}}$ of corules is called *conservative* if all of them have shape

$$\frac{c_1 \Rightarrow \langle r^1_\infty, o^1_\infty \rangle \quad \dots \quad c_n \Rightarrow \langle r^n_\infty, o^n_\infty \rangle}{c \Rightarrow \langle \infty, o_\infty \rangle}$$

We now show that, by adding a conservative set of corules to $\mathcal{R}_\infty$, we do not affect the semantics of finite computations. The intuition behind such a result is that a conservative set of corules allows infinite derivations *only* for judgements modelling divergence. The important consequence is that, for converging judgments, we can reason by standard inductive techniques on $\mathcal{R}$.

7 More precisely, O can be injectively embedded in O_∞.

THEOREM 6.31 (Conservativity): For any conservative set of corules $C^{\mathcal{R}}$, $\langle \mathcal{R}_\infty, C^{\mathcal{R}} \rangle \vdash_\nu c \Rightarrow \langle r, o \rangle$ iff $\mathcal{R} \vdash_\mu c \Rightarrow \langle r, o \rangle$.

Proof: The right-to-left implication is trivial as $\mathcal{R} \subseteq \mathcal{R}_\infty$ by Definition 6.29. To get the other direction, note that if $\langle \mathcal{R}_\infty, C^{\mathcal{R}} \rangle \vdash_\nu c \Rightarrow \langle r, o \rangle$ then we have $\mathcal{R}_\infty \cup C^{\mathcal{R}} \vdash_\mu c \Rightarrow \langle r, o \rangle$. Hence, we prove by induction on rules in $\mathcal{R}_\infty \cup C^{\mathcal{R}}$ that, if $\mathcal{R}_\infty \cup C^{\mathcal{R}} \vdash_\mu c \Rightarrow \langle r, o \rangle$, then $\mathcal{R} \vdash_\mu c \Rightarrow \langle r, o \rangle$. The cases of coaxiom $\mathsf{div}_{\mathsf{co}}(c)$ and divergence propagation $\mathsf{prop}(\rho, i, \infty)$ are both empty, as the conclusion of such rules has shape $c \Rightarrow \infty$. The only relevant case is that of a rule $\rho \in \mathcal{R}$, for which the thesis follows immediately. □

We now define the two sets of corules we will consider throughout this chapter.

DEFINITION 6.32 (Corule patterns): The set $C^{\mathcal{R}}_{\mathsf{e}}$ consists of the following coaxioms:

(CO-UNIT) for each configuration $c \in C$, define coaxiom $\text{co-unit}(c)$ as

$$\frac{}{c \Rightarrow \langle \infty, \mathsf{e} \rangle}$$

The set $C^{\mathcal{R}}_{\mathsf{g}}$ consists of the following corules:

(CO-GEN) for each rule $\rho = \mathsf{rule}(j_1 \ldots j_n,\ o_0 \ldots o_n,\ c,\ r)$ in $\mathcal{R}$, index $i \in 0..n$ such that $o_i \neq \mathsf{e}$, and possibly infinite observation $o_\infty \in O_\infty$, define $\text{co-gen}(\rho,\ i,\ o_\infty)$ as

$$\frac{j_1 \quad \ldots \quad j_i}{c \Rightarrow \langle \infty, o_0 * O(j_1) * \cdots * o_{i-1} * O(j_i) * o_i *^{\mathsf{m}} o_\infty \rangle}$$

The set $C^{\mathcal{R}}_{\mathsf{eg}}$ is defined as the union $C^{\mathcal{R}}_{\mathsf{e}} \cup C^{\mathcal{R}}_{\mathsf{g}}$.

Note that all the above defined sets of corules, namely, $C^{\mathcal{R}}_{\mathsf{e}}$, $C^{\mathcal{R}}_{\mathsf{g}}$ and $C^{\mathcal{R}}_{\mathsf{eg}}$, are conservative, hence Theorem 6.31 applies to the associated semantics.

EXAMPLE 6.33 : Due to the condition $o_i \neq \mathsf{e}$, for the λ-calculus with output (cf. Figure 6.1) the above definition associates no corule (CO-GEN) with (APP), whereas for (OUT) we obtain corule (CO-OUT), see Figure 6.4.

The properties of the semantics extension strongly depend on how the ω-monoid $C_\infty(O) = \langle O, O_\infty \rangle$ behaves with respect to the *elementary observations* $E_{\mathcal{R}}$ produced in the semantics, defined by

$$E_{\mathcal{R}} = \{o \in O \mid E(\rho, i) = o \text{ for some } \rho \in \mathcal{R} \text{ and } i \in 1..\#\rho\}$$

Further, we define $O_{\mathcal{R}}$ as the submonoid of O generated by $E_{\mathcal{R}}$. This submonoid contains all the observations produced by finite computations, as stated in the following lemma.

LEMMA 6.34 : If $\mathcal{R} \vdash_\mu c \Rightarrow \langle r, o \rangle$, then $o \in O_{\mathcal{R}}$.

Proof: The proof is by induction on rules in $\mathcal{R}$. Let $\rho = \mathsf{rule}(j_1 \dots j_n,\ o_0 \dots o_n,\ c,\ r)$ be a rule in $\mathcal{R}$, then, by Definition 6.21, we have $O(\rho) = (\prod_{i \in 1..n} o_{i-1} * O(j_i)) * o_n$. By induction hypothesis, we know that $O(j_i) \in O_{\mathcal{R}}$, for all $i \in 1..n$, and, since $o_i \in E_{\mathcal{R}}$, for all $i \in 0..n$, by definition of $E_{\mathcal{R}}$, we get the thesis, because $O_{\mathcal{R}}$ is closed under product of elements in $E_{\mathcal{R}}$. □

We are now ready to define the extension of big-step semantics, using the constructions in Definitions 6.29 and 6.32.

DEFINITION 6.35 (Extending big-step semantics): The *extension* of a big-step semantics with observations $\langle C, R, O, \mathcal{R} \rangle$ is defined by the inference system with corules $\langle \mathcal{R}_\infty, C^{\mathcal{R}}_{\mathsf{eg}} \rangle$ if $E_{\mathcal{R}}$ has unique limits in O_∞, otherwise by $\langle \mathcal{R}_\infty, C^{\mathcal{R}}_{\mathsf{e}} \rangle$.

In the above definition, if $E_{\mathcal{R}}$ has unique limits in O_∞, we take both corule patterns of Definition 6.32 and the construction is correct, as will be formally shown in Section 6.5.2 (cf. Theorems 6.40 and 6.45). Indeed, given a computation which produces infinitely many elementary non-unit observations, pattern (CO-GEN) allows *any limit product* of this sequence (cf. Lemma 6.43), hence the uniqueness of limits is needed to avoid spurious observations. The property of unique limits holds in many significant examples, see the next section, notably for the common case where observations are traces.

If this property does not hold, we can keep only pattern (CO-UNIT) and, in this way, the construction is correct for computations producing observations which are finitely generated by $E_{\mathcal{R}}$ (cf. Theorems 6.41 and 6.42), as defined above. This is satisfactory in many examples, see again the next section.

6.4 Examples of instantiation of the construction

In this section, we consider several examples, with different underlying monoids of finite observations. For simplicity, we directly show the (possibly simplified) meta-rules obtained by the construction, using the following convention: non-bold for original meta-rules, bold black for added meta-(co)rules, bold gray for extended meta-rules (merging original and added meta-rules).

6.4.1 I/O events

The first example, in Figure 6.9, is a slight extension of the λ-calculus in Section 6.1: besides `out` e, we add the construct `in` to read input values. Single observations are no longer just values, but I/O events of shape either `in` v (value v has been read) or `out` v (value v has been output). The monoid of finite observations is $\Sigma^\star_{\mathtt{io}}$, where $\Sigma_{\mathtt{io}} = \{\mathtt{in}\ v, \mathtt{out}\ v \mid v \text{ value}\}$, that is, the free monoid as in Section 6.1, but on top of a different alphabet. Hence the ω-monoid completion yields $\langle \Sigma^\star_{\mathtt{io}}, \Sigma^\infty_{\mathtt{io}} \rangle$, adding infinite sequences of events.

e	::=	$v \mid x \mid e_1\ e_2 \mid \mathtt{in} \mid \mathtt{out}\ e$	expressions
u, v	::=	$i \mid \lambda x.e$	values
θ	::=	$\mathtt{in}\ v \mid \mathtt{out}\ v$	I/O events
o	::=	$\theta_1 \dots \theta_n$	finite observations
o_∞	::=	$o \mid \theta_1 \dots \theta_n \dots$	observations
$\mathcal{D}[\ ]$	::=	$\Box\ e \mid \mathtt{out}\ \Box$	(divergence) propagation contexts

$$\textsc{(val)}\ \frac{}{\langle v, \sigma\rangle \Rightarrow \langle\langle v, \sigma\rangle, \varepsilon\rangle} \qquad \textsc{(out)}\ \frac{\langle e, \sigma_1\rangle \Rightarrow \langle\langle v, \sigma_2\rangle, o\rangle}{\langle \mathtt{out}\ e, \sigma_1\rangle \Rightarrow \langle\langle v, \sigma_2\rangle, o \cdot (\mathtt{out}\ v)\rangle}$$

$$\textsc{(in)}\ \frac{}{\langle \mathtt{in}, v\sigma\rangle \Rightarrow \langle\langle v, \sigma\rangle, \mathtt{in}\ v\rangle} \qquad \textsc{(app)}\ \frac{\begin{array}{l}\langle e_1, \sigma_1\rangle \Rightarrow \langle\langle \lambda x.e, \sigma_2\rangle, o_1\rangle \\ \langle e_2, \sigma_2\rangle \Rightarrow \langle\langle v, \sigma_3\rangle, o_2\rangle \\ \langle e[v/x], \sigma_3\rangle \Rightarrow w\end{array}}{\langle e_1\ e_2, \sigma_1\rangle \Rightarrow o_1 \cdot o_2 \cdot w}$$

$$\textsc{(div-app2)}\ \frac{\begin{array}{l}\langle e_1, \sigma_1\rangle \Rightarrow \langle\langle \lambda x.e, \sigma_2\rangle, o\rangle \\ \langle e_2, \sigma_2\rangle \Rightarrow \langle \infty, o_\infty\rangle\end{array}}{\langle e_1\ e_2, \sigma_1\rangle \Rightarrow \langle \infty, o\ \cdot o_\infty\rangle} \qquad \textsc{(div)}\ \frac{\langle e, \sigma\rangle \Rightarrow \langle \infty, o_\infty\rangle}{\langle \mathcal{D}[e], \sigma\rangle \Rightarrow \langle \infty, o_\infty\rangle}$$

$$\textsc{(co-empty)}\ \frac{}{\langle e, \sigma\rangle \Rightarrow \langle \infty, \varepsilon\rangle} \qquad \textsc{(co-in)}\ \frac{}{\langle \mathtt{in}, v\sigma\rangle \Rightarrow \langle \infty, (\mathtt{in}\ v) \cdot o_\infty\rangle}$$

$$\textsc{(co-out)}\ \frac{\langle e, \sigma_1\rangle \Rightarrow \langle\langle v, \sigma_2\rangle, o\rangle}{\langle \mathtt{out}\ e, \sigma_1\rangle \Rightarrow \langle \infty, o \cdot (\mathtt{out}\ v) \cdot o_\infty\rangle}$$

FIGURE 6.9 λ-calculus with I/O: meta-(co)rules generated by the construction

The grammar also defines (divergence) propagation contexts $\mathcal{D}[\ ]$ (Ancona, Dagnino, and Zucca, 2018) with one hole at fixed depth 1 to allow a more concise presentation of the meta-rules added for divergence propagation (see comments to rule (DIV) below).

Meta-rules (VAL), (OUT), and (IN) are original meta-rules; (VAL) and (OUT) are analogous to those in Figure 6.1. However, here configurations have shape $\langle e, \sigma\rangle$ where σ is an infinite sequence of values modeling the input stream. In meta-rule (IN), a value is read from such a stream, emitting the corresponding elementary observation.

Meta-rule (APP) is the merge of two different meta-rules: the original one, analogous to (APP) in Figure 6.1, and that added for divergence propagation from the third premise, analogous to (DIV-APP3) in Figure 6.2. To this aim, the meta-variable w ranges over pairs of shape either $\langle\langle v, \sigma\rangle, o\rangle$ or $\langle \infty, o_\infty\rangle$; accordingly, $o' \cdot w$ denotes either $\langle\langle v, \sigma\rangle, o' \cdot o\rangle$ or $\langle \infty, o' \cdot o_\infty\rangle$. Meta-rule (DIV-APP2) is analogous to that in Figure 6.2, added for divergence propagation from the second premise of (APP). Thanks to propagation contexts, the remaining meta-rule (DIV) represents those added for divergence propagation from the first premise of both (APP) and (OUT), analogously to (DIV-APP1) and (DIV-OUT) in Figure 6.2.

The meta-corules (CO-EMPTY) and (CO-OUT) are analogous to those in Figure 6.4, obtained as special cases of (CO-UNIT) and (CO-GEN) defined in Section 6.3, respectively, where the latter pattern is applied to meta-rule (OUT). The meta-corule (CO-IN) is obtained by applying the pattern (CO-GEN) to meta-rule (IN).

In this example, as in that of Section 6.1, by adding both the (CO-UNIT) and the (CO-GEN) patterns, as shown above, we get the expected semantics, that is, the same defined by the associated transition relation. Notably, completeness holds adding both patterns (Theorem 6.40), and soundness holds since the monoid of finite observations has unique limits (Theorem 6.45).

6.4.2 I/O costs

In the next example, the language is the same, but here we observe the (time) costs associated with each I/O operation. This could be easily generalized to other constructs, *e.g.*, considering also the costs for function application; however, by considering I/O operations only, we can show that our construction leads to exactly the same meta-rules as the previous example, modulo the used monoid of finite observations.

This monoid is $\langle \mathbb{R}_{\geq 0}, +, 0 \rangle$, that is, non-negative real numbers with addition. The completion yields the ω-monoid $\langle \mathbb{R}_{\geq 0}, \mathbb{R}_{\geq 0} + \{\infty\} \rangle$, that is, the only additional infinite observation is ∞, corresponding to diverging sums and with the obvious behavior with respect to the mixed product.

The meta-rules in Figure 6.10 differ from those in Figure 6.9 mainly for the employed ω-monoids, and few other details. Namely, meta-variables c and c_∞ range over $\mathbb{R}_{\geq 0}$ (finite observations) and $\mathbb{R}_{\geq 0} + \{\infty\}$ (possibly infinite observations), and the semantics is parametric in the two functions $c_{\mathtt{in}} : \mathsf{Val} \to \mathbb{R}_{\geq 0}$ and $c_{\mathtt{out}} : \mathsf{Val} \to \mathbb{R}_{\geq 0}$ assigning costs to `in` and `out` operations, respectively, depending on the input/output value. The meta-corule corresponding to the pattern (CO-UNIT) in Section 6.3 has been named (CO-ZERO). As in Figure 6.9, we overload notation by adopting the same symbol (+ in this case) for both finite and mixed product. In this case, the meta-variable w ranges over pairs of shape either $\langle \langle v, \sigma \rangle, c \rangle$ or $\langle \infty, c_\infty \rangle$; accordingly, $c' + w$ denotes either $\langle \langle v, \sigma \rangle, c' + c \rangle$ or $\langle \infty, c' + c_\infty \rangle$.

As in the previous example, by adding both the (CO-UNIT) and (CO-GEN) patterns we get the expected semantics. Completeness is again ensured by Theorem 6.40. Soundness (Theorem 6.45) holds under the following assumption on the cost functions: $0 < \inf\{c_{\mathtt{in}}(v), c_{\mathtt{out}}(v) \mid v \in \mathsf{Val}, 0 < c_{\mathtt{in}}(v), 0 < c_{\mathtt{out}}(v)\}$; that is, non-zero costs for I/O operations cannot be arbitrarily close to zero. This ensures that the set of elementary observations produced in the semantics has unique limits. This is a reasonable assumption: it means that diverging programs performing infinitely many I/O operations with non-zero costs cannot have a finite cost.

6.4.3 Executed branches

We consider a λ-calculus with labelled conditional expressions and boolean constants, and a semantics useful to reason about branch coverage. The syntax is defined in the top section of Figure 6.11.

$$\text{(VAL)}\ \frac{}{\langle v,\sigma\rangle \Rightarrow \langle\langle v,\sigma\rangle, 0\rangle} \qquad \text{(OUT)}\ \frac{\langle e,\sigma_1\rangle \Rightarrow \langle\langle v,\sigma_2\rangle, c\rangle}{\langle \mathtt{out}\ e,\sigma_1\rangle \Rightarrow \langle\langle v,\sigma_2\rangle, c + c_{\mathtt{out}}(v)\rangle}$$

$$\text{(IN)}\ \frac{}{\langle \mathtt{in}, v\sigma\rangle \Rightarrow \langle\langle v,\sigma\rangle, c_{\mathtt{in}}(v)\rangle} \qquad \text{(APP)}\ \frac{\begin{array}{l}\langle e_1,\sigma_1\rangle \Rightarrow \langle\langle \lambda x.e,\sigma_2\rangle, c_1\rangle\\ \langle e_2,\sigma_2\rangle \Rightarrow \langle\langle v,\sigma_3\rangle, c_2\rangle\\ \langle e[v/x],\sigma_3\rangle \Rightarrow w\end{array}}{\langle e_1\ e_2,\sigma_1\rangle \Rightarrow c_1 + c_2 + w}$$

$$\text{(DIV-APP2)}\ \frac{\begin{array}{l}\langle e_1,\sigma_1\rangle \Rightarrow \langle\langle \lambda x.e,\sigma_2\rangle, c\rangle\\ \langle e_2,\sigma_2\rangle \Rightarrow \langle\infty, c_\infty\rangle\end{array}}{\langle e_1\ e_2,\sigma_1\rangle \Rightarrow \langle\infty, c + c_\infty\rangle} \qquad \text{(DIV)}\ \frac{\langle e,\sigma\rangle \Rightarrow \langle\infty, c_\infty\rangle}{\langle \mathcal{D}[e],\sigma\rangle \Rightarrow \langle\infty, c_\infty\rangle}$$

$$\text{(CO-ZERO)}\ \frac{}{\langle e,\sigma\rangle \Rightarrow \langle\infty, 0\rangle} \qquad \text{(CO-IN)}\ \frac{}{\langle \mathtt{in}, v\sigma\rangle \Rightarrow \langle\infty, c_{\mathtt{in}}(v) + c_\infty\rangle}$$

$$\text{(CO-OUT)}\ \frac{\langle e,\sigma_1\rangle \Rightarrow \langle\langle v,\sigma_2\rangle, c\rangle}{\langle \mathtt{out}\ e,\sigma_1\rangle \Rightarrow \langle\infty, c + c_{\mathtt{out}}(v) + c_\infty\rangle}$$

FIGURE 6.10 λ-calculus with I/O costs: meta-(co)rules generated by the construction

We assume each conditional expression $e\ ?_a\ e_1\ :\ e_2$ in a program to be associated with a unique label a ranging over a countably infinite set $\mathcal{A}$ of labels, so that $a.\mathtt{true}$ and $a.\mathtt{false}$ denote the unique addresses inside the program of the *then* and *else* branches e_1 and e_2, respectively.

Here finite observations are the sets of the addresses of the branches executed by a program, represented by the monoid $\langle \wp_\omega(\mathcal{A}_{\mathsf{br}}), \cup, \emptyset\rangle$, where $\mathcal{A}_{\mathsf{br}} = \{a.\mathtt{true}, a.\mathtt{false} \mid a \in \mathcal{A}\}$. The completion yields the ω-monoid $\langle \wp_\omega(\mathcal{A}_{\mathsf{br}}), \wp(\mathcal{A}_{\mathsf{br}})\rangle$ (see Example 6.3 (4)), adding infinite subsets of $\mathcal{A}_{\mathsf{br}}$. Again, in the meta-rules, the symbol $\cup$ denotes both the finite and the mixed product. The meta-variable w ranges over pairs of shape either $\langle v, A\rangle$ or $\langle\infty, A_\infty\rangle$, and, if $w = \langle v_\infty, A_\infty\rangle$, then $A \cup w$ denotes $\langle v_\infty, A \cup A_\infty\rangle$.

Similarly to meta-rule (APP), meta-rules (IF-F) and (IF-T) (name in gray bold) are obtained by merging two different meta-rules: the original one for the finite semantics of conditional expressions, and the meta-rule representing those added for divergence propagation from the second premise.

For what concerns meta-corules, in this example we add only the (CO-UNIT) pattern, because adding (CO-GEN) would be unsound. Indeed, as already mentioned, the completion produces the full powerset $\wp(\mathcal{A}_{\mathsf{br}})$, and the set of elementary observations has not unique limits. However, since every program has a finite set of branches, it is easy to see that (even infinite) computations in the associated transition system produce only observations which are finitely generated by elementary observations, hence Theorem 6.41 gives completeness of the big-step semantics. And, because we have only the (CO-UNIT) pattern, by Theorem 6.42 the big-step semantics is sound.

$$
\begin{array}{rcll}
e & ::= & v \mid x \mid e_1\, e_2 \mid e\, ?_a\, e_1 : e_2 & \text{expressions} \\
u, v & ::= & \texttt{true} \mid \texttt{false} \mid \lambda x.e & \text{values} \\
\mathcal{D}[\,] & ::= & \Box\, e \mid \Box\, ?_a\, e_1 : e_2 & \text{propagation contexts}
\end{array}
$$

$$
(\textsc{val})\ \frac{}{v \Rightarrow \langle v, \emptyset \rangle} \qquad (\textsc{app})\ \frac{\begin{array}{c} e_1 \Rightarrow \langle \lambda x.e, A_1 \rangle \\ e_2 \Rightarrow \langle v, A_2 \rangle \\ e[v/x] \Rightarrow w \end{array}}{e_1\, e_2 \Rightarrow A_1 \cup A_2 \cup w}
$$

$$
(\textsc{if-f})\ \frac{\begin{array}{c} e \Rightarrow \langle \texttt{false}, A \rangle \\ e_2 \Rightarrow w \end{array}}{e\, ?_a\, e_1 : e_2 \Rightarrow A \cup \{a.\texttt{false}\} \cup w} \qquad (\textsc{if-t})\ \frac{\begin{array}{c} e \Rightarrow \langle \texttt{true}, A \rangle \\ e_1 \Rightarrow w \end{array}}{e\, ?_a\, e_1 : e_2 \Rightarrow A \cup \{a.\texttt{true}\} \cup w}
$$

$$
(\textsc{div-app2})\ \frac{\begin{array}{c} e_1 \Rightarrow \langle \lambda x.e, A \rangle \\ e_2 \Rightarrow \langle \infty, A_\infty \rangle \end{array}}{e_1\, e_2 \Rightarrow \langle \infty, A \cup A_\infty \rangle} \qquad (\textsc{div})\ \frac{e \Rightarrow \langle \infty, A_\infty \rangle}{\mathcal{D}[e] \Rightarrow \langle \infty, A_\infty \rangle}
$$

$$
(\textsc{co-empty})\ \frac{}{e \Rightarrow \langle \infty, \emptyset \rangle}
$$

FIGURE 6.11 λ-calculus with conditional: meta-(co)rules generated by the construction

6.4.4 Maximum heap size

In this last example we consider an imperative extension of the call-by-value λ-calculus with heap references which can be explicitly deallocated. The syntax is in the top section of Figure 6.12.

Values are either references ι or λ-abstractions. The syntax includes expressions of shape $\texttt{ref}\ e$ creating a new reference initialized with the value of e, $!\ e$ dereferencing the reference denoted by e, $e_1{=}e_2$ updating the reference denoted by e_1 with the value of e_2, and $\texttt{free}\ e$ deallocating the reference denoted by e.

In this case we are interested in observing the maximum size of the heap used by a program: finite observations are modelled by the monoid $\langle \mathbb{N}, \vee, 0 \rangle$. This monoid can be employed whenever observing the maximum number of used resources, independently from the notion of resource (heap locations, files, locks, etc.). The completion yields the ω-monoid $\langle \mathbb{N}, \mathbb{N} + \{\infty\} \rangle$ (see Example 6.3 (3)), whose infinite product computes the supremum of values in a given sequence.

As before, the same symbol $\vee$ denotes both the finite and the mixed product. The meta-variable w ranges over pairs of shape either $\langle \langle v, \mathcal{H} \rangle, s \rangle$ or $\langle \infty, s_\infty \rangle$; accordingly, $s' \vee w$ denotes either $\langle \langle v, \mathcal{H} \rangle, s' \vee s \rangle$ or $\langle \infty, s' \vee s_\infty \rangle$.

Configurations have shape $\langle e, \mathcal{H} \rangle$, where a heap $\mathcal{H}$ is a finite map from references to values. Heap extension is denoted by $\mathcal{H} \uplus \{\iota \mapsto v\}$ (where ι is not in the domain of $\mathcal{H}$); $|\mathcal{H}|$ denotes the cardinality of the domain of $\mathcal{H}$, *i.e.*, its size[8]. If the computation converges, then $\langle \langle v, \mathcal{H} \rangle, s \rangle$ is returned (a value v, a heap $\mathcal{H}$, and a maximum size s); if it diverges, then a pair $\langle \infty, s_\infty \rangle$ is returned.

8 With the simplifying assumption that the size does not depend on the values in the heap.

$$
\begin{array}{rcll}
e & ::= & v \mid x \mid e_1\ e_2 \mid \texttt{ref}\ e \mid \texttt{!}\ e \mid e_1\texttt{=}e_2 \mid \texttt{free}\ e & \text{expressions} \\
v & ::= & \iota \mid \lambda x.e & \text{values} \\
\mathcal{D}[\,] & ::= & \Box\ e \mid \texttt{ref}\ \Box \mid \texttt{!}\ \Box \mid \texttt{free}\ \Box \mid \Box\texttt{=}e & \text{propagation contexts}
\end{array}
$$

$$
\textsc{(val)}\ \frac{}{\langle v, \mathcal{H}\rangle \Rightarrow \langle\langle v, \mathcal{H}\rangle, 0\rangle} \qquad \textsc{(ref)}\ \frac{\langle e, \mathcal{H}_1\rangle \Rightarrow \langle\langle v, \mathcal{H}_2\rangle, s\rangle}{\langle \texttt{ref}\ e, \mathcal{H}_1\rangle \Rightarrow \langle\langle \iota, \mathcal{H}_2 \uplus \{\iota \mapsto v\}\rangle, s \vee (1 + |\mathcal{H}_2|)\rangle}
$$

$$
\textsc{(deref)}\ \frac{\langle e, \mathcal{H}_1\rangle \Rightarrow \langle\langle \iota, \mathcal{H}_2\rangle, s\rangle}{\langle \texttt{!}\ e, \mathcal{H}_1\rangle \Rightarrow \langle\langle v, \mathcal{H}_2\rangle, s \vee |\mathcal{H}_2|\rangle}\ \mathcal{H}_2(\iota) = v
$$

$$
\textsc{(free)}\ \frac{\langle e, \mathcal{H}_1\rangle \Rightarrow \langle\langle \iota, \mathcal{H}_2\rangle, s\rangle}{\langle \texttt{free}\ e, \mathcal{H}_1\rangle \Rightarrow \langle\langle v, \mathcal{H}_3\rangle, s \vee |\mathcal{H}_3|\rangle}\ \mathcal{H}_2 = \mathcal{H}_3 \uplus \{\iota \mapsto v\}
$$

$$
\textsc{(upd)}\ \frac{\langle e_1, \mathcal{H}_1\rangle \Rightarrow \langle\langle \iota, \mathcal{H}_2\rangle, s_1\rangle \quad \langle e_2, \mathcal{H}_2\rangle \Rightarrow \langle\langle v, \mathcal{H}_3\rangle, s_2\rangle}{\langle e_1\texttt{=}e_2, \mathcal{H}_1\rangle \Rightarrow \langle\langle v, \mathcal{H}_2 \uplus \{\iota \mapsto v\}\rangle, s_1 \vee s_2 \vee |\mathcal{H}_3|\rangle}\ \mathcal{H}_3 = \mathcal{H}_2 \uplus \{\iota \mapsto v'\}
$$

$$
\textsc{(app)}\ \frac{\langle e_1, \mathcal{H}_1\rangle \Rightarrow \langle\langle \lambda x.e, \mathcal{H}_2\rangle, s_1\rangle \quad \langle e_2, \mathcal{H}_2\rangle \Rightarrow \langle\langle v, \mathcal{H}_3\rangle, s_2\rangle \quad \langle e[v/x], \mathcal{H}_3\rangle \Rightarrow w}{\langle e_1\ e_2, \mathcal{H}_1\rangle \Rightarrow s_1 \vee s_2 \vee |\mathcal{H}_3| \vee w}
$$

$$
\textsc{(div-upd)}\ \frac{\langle e_1, \mathcal{H}_1\rangle \Rightarrow \langle\langle \iota, \mathcal{H}_2\rangle, s\rangle \quad \langle e_2, \mathcal{H}_2\rangle \Rightarrow \langle \infty, s_\infty\rangle}{\langle e_1\texttt{=}e_2, \mathcal{H}_1\rangle \Rightarrow \langle \infty, s \vee s_\infty\rangle} \qquad \textsc{(div)}\ \frac{\langle e, \mathcal{H}\rangle \Rightarrow \langle \infty, s_\infty\rangle}{\langle \mathcal{D}[e], \mathcal{H}\rangle \Rightarrow \langle \infty, s_\infty\rangle}
$$

$$
\textsc{(div-app2)}\ \frac{\langle e_1, \mathcal{H}_1\rangle \Rightarrow \langle\langle \lambda x.e, \mathcal{H}_2\rangle, s\rangle \quad \langle e_2, \mathcal{H}_2\rangle \Rightarrow \langle \infty, s_\infty\rangle}{\langle e_1\ e_2, \mathcal{H}_1\rangle \Rightarrow \langle \infty, s \vee s_\infty\rangle} \qquad \textsc{(co-zero)}\ \frac{}{\langle e, \mathcal{H}\rangle \Rightarrow \langle \infty, 0\rangle}
$$

FIGURE 6.12 λ-calculus with references: meta-(co)rules generated by the construction

As in the previous example, the set of elementary observations produced in the semantics has not unique limits. Hence, we keep only the meta-coaxiom (co-zero) corresponding to the pattern (co-unit) to avoid unsoundness. The big-step semantics is sound by Theorem 6.42; however, as opposed to the previous example, the infinitely generated observation[9] ∞ is not obtained, hence the semantics is not complete. However, by Theorem 6.41 the semantics is complete for finitely generated observations; since the only infinitely generated observation is ∞, the only case where the big-step semantics does not return any result is for programs that would require an infinite heap to run. This is acceptable, as such programs are always doomed to crash.

6.5 Correctness of the construction

In this section we prove the correctness of the construction in Section 6.3.3. To formulate and prove correctness, we use as reference semantics the one defined in Section 6.3.2 through the labelled transition relation $\longrightarrow_{\mathcal{R}}$, where the definition of both finite and infinite computations is straightforward. Correctness then means that, starting from a big-step semantics $\langle C, R, O, \mathcal{R}\rangle$, through the construction we get an extended big-step semantics equivalent to possibly infinite computations in $\longrightarrow_{\mathcal{R}}$.

9 Produced by diverging programs allocating infinitely many references, without deallocating them.

Throughout this section we assume a big-step semantics with observations $\langle C, R, O, \mathcal{R}\rangle$, where the completion of O is $C_\infty(O) = \langle O, O_\infty\rangle$. First of all, we introduce notations for the semantics derived by the transition relation $\longrightarrow_{\mathcal{R}}$:

- $c \rightsquigarrow_{\mathcal{R}} \langle r, o\rangle$ means that there is a finite computation starting from c producing the result r, with observation o, obtained interpreting the finite sequence of observations of single steps. Formally, we define

  $$c \rightsquigarrow_{\mathcal{R}} \langle r, o\rangle \text{ iff } \frac{}{c \Rightarrow ?} \xrightarrow{u}{}^{\star}_{\mathcal{R}} \tau,\ R(\mathsf{r}(\tau)) = r \text{ and } \mathsf{it}(u) = o \text{ for some } u \in O^\star$$

- $c \rightsquigarrow_{\mathcal{R}} \langle \infty, o_\infty\rangle$ means that there is an infinite computation starting from c, with possibly infinite observation o_∞, obtained interpreting the infinite sequence of observations of single steps. Formally, let us define $\tau \xrightarrow{\sigma}{}^{\omega}_{\mathcal{R}}$, where τ is a finite partial evaluation tree and $\sigma \in O^\omega$, as the coinductive interpretation of the following rule:

  $$\frac{\tau' \xrightarrow{\sigma}{}^{\omega}_{\mathcal{R}} \quad \tau \xrightarrow{o}_{\mathcal{R}} \tau'}{\tau \xrightarrow{o\sigma}{}^{\omega}_{\mathcal{R}}}$$

  That is, $\tau \xrightarrow{\sigma}{}^{\omega}_{\mathcal{R}}$, with $\sigma = (o_i)_{i\in\mathbb{N}}$, iff there is an infinite sequence $(\tau_i)_{i\in\mathbb{N}}$ such that $\tau = \tau_0$ and, for all $i \in \mathbb{N}$, $\tau_i \xrightarrow{o_i}_{\mathcal{R}} \tau_{i+1}$. Then, we define

  $$c \rightsquigarrow_{\mathcal{R}} \langle \infty, o_\infty\rangle \text{ iff } \frac{}{c \Rightarrow ?} \xrightarrow{\sigma}{}^{\omega}_{\mathcal{R}} \text{ and } \mathsf{it}^\infty(\sigma) = o_\infty \text{ for some } \sigma \in O^\omega.$$

The equivalence for finite computations is an easy consequence of Theorems 6.28 and 6.31.

THEOREM 6.36 (Equivalence for finite computations): Let $C^{\mathcal{R}}$ be a conservative set of corules, then $\langle \mathcal{R}_\infty, C^{\mathcal{R}}\rangle \vdash_\nu c \Rightarrow \langle r, o\rangle$ iff $c \rightsquigarrow_{\mathcal{R}} \langle r, o\rangle$.

Proof: By Theorem 6.31, $\langle \mathcal{R}, C^{\mathcal{R}}\rangle \vdash_\nu c \Rightarrow \langle r, o\rangle$ is equivalent to $\mathcal{R} \vdash_\mu c \Rightarrow \langle r, o\rangle$ and, by Theorem 6.28, this is equivalent to $\frac{}{c \Rightarrow ?} \xrightarrow{u}{}^{\star}_{\mathcal{R}} \tau$, for some $u \in O^\star$, where $\mathsf{r}(\tau) = c \Rightarrow \langle r, o\rangle$ and $\mathsf{it}(u) = o$, hence we have the thesis. □

The correctness result for infinite computations requires an additional assumption on big-step semantics: we assume that it is *deterministic*. Such an assumption needs to be expressed, rather than globally on the semantic relation, at the level of individual rules, since the property should be preserved by the construction, which handles single rules.

6.5.1 Determinism assumptions

Determinism is expressed at the level of single rules as follows:

ASSUMPTION 6.2 : For all

$$\rho = \mathsf{rule}(j_1 \ldots j_n,\ o_0 \ldots o_n,\ c,\ r) \text{ and } \rho' = \mathsf{rule}(j'_1 \ldots j'_m,\ o'_0 \ldots o'_m,\ c',\ r')$$

in $\mathcal{R}$, with $c = c'$ the following hold:

1. for all $i \in 1..\min\{n, m\}$, if, for all $k < i$, $j_k = j'_k$, then $\rho \sim_i \rho'$, and
2. if, for all $k \in 1..\min\{n, m\}$, $j_k = j'_k$, then $\rho = \rho'$.

To explain how the above assumption actually models determinism, let us consider the (meta-)rule for application in a λ-calculus where configurations are pairs $\langle e, \mu\rangle$ with μ auxiliary structure, *e.g.*, memory, modified by some constructs, so that the evaluation order is relevant.

$$(\textsc{app})\ \frac{\langle e_1, \mu\rangle \Rightarrow \langle\langle \lambda x.e, \mu_1\rangle, o_1\rangle \quad \langle e_2, \mu_1\rangle \Rightarrow \langle\langle v, \mu_2\rangle, o_2\rangle \quad \langle e[v/x], \mu_2\rangle \Rightarrow \langle\langle u, \mu'\rangle, o\rangle}{\langle e_1\ e_2, \mu\rangle \Rightarrow \langle\langle u, \mu'\rangle, o_1 \cdot o_2 \cdot \varepsilon \cdot o\rangle}$$

Note that, for a fixed $\langle e_1\ e_2, \mu\rangle$ in the conclusion, there are infinitely many rules which can be obtained by instantiating the meta-variables. Assumption 6.2 (1) imposes the following constraints, expressed in the meta-rule by using the same meta-variable:

- ($i = 1$) the configuration in the first premise is uniquely determined
- ($i = 2$) the configuration in the second premise is uniquely determined by the result of the first premise
- ($i = 3$) the configuration in the third premise is uniquely determined by the results of the first two premises.

In (app) all elementary observations are equal to e, but, in general, Assumption 6.2 (1) requires each elementary observation to be uniquely determined by previous premises as well. Finally, Assumption 6.2 (2) requires the final result and observation to be uniquely determined by the results in the premises.

We prove now two lemmas, holding under Assumption 6.2, used in later proofs. The first states that the big-step semantic relation is indeed deterministic, that is, any configuration has at most one final result and observation. Note that, by conservativity (cf. Theorem 6.31), this ensures that any conservative set of corules preserves determinism for finite computations. Another source of non-determinism could be a conflict between convergence and divergence, and the second lemma states that this is prevented as well.

LEMMA 6.37 : If $\mathcal{R} \vdash_\mu c \Rightarrow \langle r_1, o_1\rangle$ and $\mathcal{R} \vdash_\mu c \Rightarrow \langle r_2, o_2\rangle$, then $r_1 = r_2$ and $o_1 = o_2$.

Proof: The proof is by induction on the derivation of $c \Rightarrow \langle r_1, o_1\rangle$, and we denote by *RH* the induction hypothesis. We know that $c \Rightarrow \langle r_1, o_1\rangle$ is derived by rule $\rho_1 = \mathsf{rule}(j_1 \ldots j_n,\ o_0 \ldots o_n,\ c,\ r_1)$ and $c \Rightarrow \langle r_2, o_2\rangle$ is derived by rule $\rho_2 = \mathsf{rule}(j'_1 \ldots j'_m,\ o'_0 \ldots o'_m,\ c,\ r_2)$. We prove, by complete arithmetic induction, that, for all $k \in 1..\min\{n, m\}$, $j_k = j'_k$. Let $k \in 1..\min\{n, m\}$, then, by induction hypothesis, we know that, for all $h < k$, $j_h = j'_h$, hence, by Assumption 6.2 (1), we get $\rho_1 \sim_k \rho_2$ and, in particular, $C(j_k) = C(j'_k)$. Then, by *RH*, we get $R(j_k) = R(j'_k)$ and $O(j_k) = O(j'_k)$, that is, $j_k = j'_k$.

Finally, since for all $k \in 1..\min\{n,m\}$ we have $j_k = j'_k$, by Assumption 6.2 (2) we get $\rho_1 = \rho_2$, hence $r_1 = r_2$ and $o_1 = o_2$. □

LEMMA 6.38 : If $\mathcal{R} \vdash_\mu c \Rightarrow \langle r, o\rangle$, then there is no $o_\infty \in O_\infty$ such that $\mathcal{R}_\infty \vdash_\nu c \Rightarrow \langle \infty, o_\infty\rangle$.

Proof: The proof is by induction on rules in $\mathcal{R}$. Suppose $c \Rightarrow \langle r, o\rangle$ is derived by $\rho = \mathsf{rule}(j_1 \dots j_n,\ o_0 \dots o_n,\ c,\ r)$. Assume now that $\mathcal{R}_\infty \vdash_\nu c \Rightarrow \langle \infty, o_\infty\rangle$ for some $o_\infty \in O_\infty$, then we have applied a rule $\mathsf{div}(\rho',\ i,\ o'_\infty)$ where $\rho' = \mathsf{rule}(j'_1 \dots j'_m,\ o'_0 \dots o'_m,\ c,\ r')$. We prove, by complete arithmetic induction, that, for all $k \in 1..\min\{i,n\}$, $C(j_k) = C(j'_k)$. Let $k \in 1..\min\{i,n\}$, then, by induction hypothesis, we have, for all $h < k$, $C(j_h) = C(j'_h)$, and, by Lemma 6.37, we get $j_h = j'_h$. Thus, by Assumption 6.2 (1), we get $\rho \sim_k \rho'$ and, in particular, $C(j_k) = C(j'_k)$.

We have $\min\{i,n\} = i$ or $\min\{i,n\} = n$, but in the latter case, since we have just proved $C(j_k) = C(j'_k)$, for all $k \in 1..n$, by Lemma 6.37 we get $j_k = j'_k$, hence, by Assumption 6.2 (2), we get $\rho = \rho'$; thus, we have $i \leq m = n = \min\{i,n\}$, hence $i = n$. Therefore, in both cases we have $\min\{i,n\} = i$.

Then, in particular, we have $C(j_i) = C(j'_i)$ and, by hypothesis, we have $\mathcal{R}_\infty \vdash_\nu C(j'_i) \Rightarrow \langle \infty, o'_\infty\rangle$, which is not possible by induction hypothesis, since $\mathcal{R} \vdash_\mu j_i$. □

COROLLARY 6.39 : Let $C^{\mathcal{R}}$ be a conservative set of corules. If $\langle \mathcal{R}_\infty, C^{\mathcal{R}}\rangle \vdash_\nu c \Rightarrow \langle r, o\rangle$, then there is no $o_\infty \in O_\infty$ such that $\langle \mathcal{R}_\infty, C^{\mathcal{R}}\rangle \vdash_\nu c \Rightarrow \langle \infty, o_\infty\rangle$.

Proof: By Theorem 6.31, we have $\mathcal{R} \vdash_\mu c \Rightarrow \langle r, o\rangle$ and, by Lemma 6.38, there is no $o_\infty \in O_\infty$ such that $\mathcal{R}_\infty \vdash_\nu c \Rightarrow \langle \infty, o_\infty\rangle$ and, since $\langle \mathcal{R}_\infty, C^{\mathcal{R}}\rangle \vdash_\nu c \Rightarrow \langle \infty, o_\infty\rangle$ implies $\mathcal{R}_\infty \vdash_\nu c \Rightarrow \langle \infty, o_\infty\rangle$, we get the thesis. □

6.5.2 Results for infinite computations

We now present the formal results about the correctness for infinite computations. As already mentioned, we assume the big-step semantics to satisfy Assumption 6.2.

First we discuss *completeness* for infinite computations. The following theorem states completeness for $C^{\mathcal{R}}_{\mathsf{eg}}$, that is, the extension obtained by both the (CO-UNIT) and (CO-GEN) patterns.

THEOREM 6.40 (Completeness): If $c \rightsquigarrow_{\mathcal{R}} \langle \infty, o_\infty\rangle$, then $\langle \mathcal{R}_\infty, C^{\mathcal{R}}_{\mathsf{eg}}\rangle \vdash_\nu c \Rightarrow \langle \infty, o_\infty\rangle$.

An additional completeness result characterizes infinite computations which can be derived by restricting corules to the set $C^{\mathcal{R}}_{\mathsf{e}}$ of those of shape (CO-UNIT). Recall from Section 6.3 that $E_{\mathcal{R}}$ is the set of the elementary observations produced in $\mathcal{R}$, $O_{\mathcal{R}}$ the submonoid of O generated by $E_{\mathcal{R}}$, and observations finitely generated (by $E_{\mathcal{R}}$) are those with a non-empty and finite set of factors

in $O_{\mathcal{R}}$ (see Definition 6.13). Then, the following theorem states that all infinite computations producing a finitely generated observation can be derived using only (CO-UNIT).

THEOREM 6.41 (Completeness for $C_{\mathsf{e}}^{\mathcal{R}}$): If $c \rightsquigarrow_{\mathcal{R}} \langle \infty, o_\infty \rangle$ and o_∞ is finitely generated by $E_{\mathcal{R}}$, then $\langle \mathcal{R}_\infty, C_{\mathsf{e}}^{\mathcal{R}} \rangle \vdash_\nu c \Rightarrow \langle \infty, o_\infty \rangle$.

We now discuss *soundness* results for the constructions in Definition 6.35. Soundness always holds when restricting corules to the set $C_{\mathsf{e}}^{\mathcal{R}}$, and in such case, as stated above, completeness can be kept as well if the produced observations are finitely generated, see the example in Section 6.4.3.

THEOREM 6.42 (Soundness for $C_{\mathsf{e}}^{\mathcal{R}}$): If $\langle \mathcal{R}_\infty, C_{\mathsf{e}}^{\mathcal{R}} \rangle \vdash_\nu c \Rightarrow \langle \infty, o_\infty \rangle$, then $c \rightsquigarrow_{\mathcal{R}} \langle \infty, o_\infty \rangle$.

Let us now consider the pattern (CO-GEN). We formally motivate that, as shown by the examples of Section 6.4, in this case soundness requires unique limits (cf. Definition 6.13).

Let us consider an (annotated) evaluation tree τ in $\mathcal{R}_\infty$ of the judgement $c \Rightarrow \langle \infty, o_\infty \rangle$, which is necessarily infinite, as there are no axioms in $\mathcal{R}_\infty$ introducing ∞ (cf. Definition 6.29). We can associate with τ the infinite sequence σ_τ of finite observations produced by such tree. Formally, setting $c_0 = c$, for each level i of τ there is a divergence propagation rule $\mathsf{div}(\rho_i, k_i, o_\infty^{i+1})$ with $\rho_i = \mathsf{rule}(j_{i,1} \dots j_{i,n_i}, o_{i,0} \dots o_{i,n_i}, c_i, r_i)$, $k_i \in 1..n_i$ and $o_\infty^i = o_{i,0} * (\prod_{h \in 1..k_i-1} O(j_{i,h}) * o_{i,h}) *^{\mathsf{m}} o_\infty^{i+1}$, that is, setting $o_\infty^0 = o_\infty$, τ can be described by a sequence $(\tau_i)_{i \in \mathbb{N}}$ of trees such that

$$\tau_0 = \tau \qquad \tau_i = \frac{\tau_{i,1} \ \dots \tau_{i,k_i-1} \ \tau_{i+1}}{c_i \Rightarrow \langle \infty, o_\infty^i \rangle}\,(\mathsf{div}(\rho_i, k_i, o_\infty^{i+1}))$$

where, for all $h \in 1..k_i - 1$, $\tau_{i,h}$ can be assumed finite by Theorem 6.31. We can then define $\sigma_\tau \in O^\omega$ as the sequence $(\sigma_\tau(i))_{i \in \mathbb{N}}$ where

$$\sigma_\tau(i) = o_{i,0} * \left(\prod_{h \in 1..k_i-1} O(j_{i,h}) * o_{i,h} \right)$$

Note that, for all $i \in \mathbb{N}$, $o_\infty^i = \sigma_\tau(i) *^{\mathsf{m}} o_\infty^{i+1}$, τ_i is an evaluation tree for $c_i \Rightarrow \langle \infty, o_\infty^{i+1} \rangle$ and $\sigma_{\tau_i}(k) = \sigma_\tau(i + k)$. Moreover, o_∞ is a limit product of σ_τ (cf. Definition 6.6), that is, there is a sequence $(o_\infty^n)_{n \in \mathbb{N}}$ such that, $o_\infty = o_\infty^0$ and, for all $n \in \mathbb{N}$, $o_\infty^n = \sigma_\tau(n) *^{\mathsf{m}} o_\infty^{n+1}$, which implies $o_\infty = \mathsf{it}(\sigma_\tau[n]) *^{\mathsf{m}} o_\infty^n$. Actually we have the following result.

LEMMA 6.43 : If $\langle \mathcal{R}_\infty, C_{\mathsf{eg}}^{\mathcal{R}} \rangle \vdash_\nu c \Rightarrow \langle \infty, o_\infty \rangle$ holds by an infinite evaluation tree τ, σ_τ is non-trivial and o'_∞ is a limit product of σ_τ, then $\langle \mathcal{R}_\infty, C_{\mathsf{eg}}^{\mathcal{R}} \rangle \vdash_\nu c \Rightarrow \langle \infty, o'_\infty \rangle$.

This lemma shows why adding the pattern (CO-GEN) could be not sound: for each infinite evaluation tree τ for a judgement $c \Rightarrow \langle \infty, o_\infty \rangle$, where σ_τ is non-trivial, we can derive $c \Rightarrow \langle \infty, o'_\infty \rangle$ for all limit products o'_∞ of σ_τ. Hence,

an easy sufficient condition to ensure soundness is to require such limit to be unique.

By Lemma 6.34, given an infinite evaluation tree τ of $c \Rightarrow \langle \infty, o_\infty \rangle$ in $\mathcal{R}_\infty$, σ_τ belongs to $O^\omega_\mathcal{R}$. Then, for soundness it is enough that $O_\mathcal{R}$ (in fact, thanks to the next proposition, $E_\mathcal{R}$), has unique limits.

PROPOSITION 6.44 : $O_\mathcal{R}$ has unique limits if and only if $E_\mathcal{R}$ has unique limits.

THEOREM 6.45 (Soundness): If $\langle \mathcal{R}_\infty, C^\mathcal{R}_{\mathsf{eg}} \rangle \vdash_\nu c \Rightarrow \langle \infty, o_\infty \rangle$ and $E_\mathcal{R}$ has unique limits, then $c \rightsquigarrow_\mathcal{R} \langle \infty, o_\infty \rangle$.

6.5.3 Proofs

In this subsection, we provide proofs of results for infinite computations presented in the previos subsection.

COMPLETENESS The proofs of all our completeness result rely on the following coinductive proof principle, similar to Lemma 5.28, derived from the bounded coinduction principle associated with inference systems with corules.

LEMMA 6.46 : Let $\mathcal{S} \subseteq C \times O_\infty$ be a set. If, for all $\langle c, o_\infty \rangle \in \mathcal{S}$, we have

1. $\mathcal{R}_\infty \cup C^\mathcal{R} \vdash_\mu c \Rightarrow \langle \infty, o_\infty \rangle$, and
2. there are a rule $\rho = \mathsf{rule}(j_1 \dots j_n,\ o_0 \dots o_n,\ c,\ r)$ in $\mathcal{R}$, an index $i \in 1..n$ and $o'_\infty \in O_\infty$, such that
   - $o_\infty = (\prod_{k \in 1..i-1} o_{k-1} * O(j_k)) * o_{i-1} *^{\mathsf{m}} o'_\infty$,
   - for all $k < i$, $\mathcal{R} \vdash_\mu j_k$ and
   - $\langle C(j_i), o'_\infty \rangle \in \mathcal{S}$,

then, for all $\langle c, o_\infty \rangle \in \mathcal{S}$, $\langle \mathcal{R}_\infty, C^\mathcal{R} \rangle \vdash_\nu c \Rightarrow \langle \infty, o_\infty \rangle$.

Proof: Consider the set $\mathcal{S}' = \{\langle c, \infty, o_\infty \rangle \mid \langle c, o_\infty \rangle \in \mathcal{S}\} \cup \{\langle c, r, o \rangle \mid \mathcal{R} \vdash_\mu c \Rightarrow \langle r, o \rangle\}$, then the proof is by bounded coinduction (cf. Proposition 3.27).

BOUNDEDNESS We have to show that, for all $\langle c, r_\infty, o_\infty \rangle \in \mathcal{S}'$, $\mathcal{R}_\infty \cup C^\mathcal{R} \vdash_\mu c \Rightarrow \langle r_\infty, o_\infty \rangle$ holds. This is easy because, if $r_\infty = \infty$, then this holds by hypothesis (Item 1), otherwise $r_\infty \in R$, $o_\infty \in O$ and $\mathcal{R} \vdash_\mu c \Rightarrow \langle r_\infty, o_\infty \rangle$, hence this holds since $\mathcal{R} \subseteq \mathcal{R}_\infty \subseteq \mathcal{R}_\infty \cup C^\mathcal{R}$.

CONSISTENCY We have to show that, for all $\langle c, r_\infty, o_\infty \rangle \in \mathcal{S}'$, there is a rule $\langle j_1 \dots j_n, c \Rightarrow \langle r_\infty, o_\infty \rangle \rangle \in \mathcal{R}_\infty$ such that, for all $k \in 1..n$, $\langle C(j_k), R_\infty(j_k), O_\infty(j_k) \rangle \in \mathcal{S}'$. There are two cases:

- If $r_\infty = \infty$, then by hypothesis (Item 2), we have a rule $\rho = \mathsf{rule}(j_1 \dots j_n,\ o_0 \dots o_n,\ c,\ r) \in \mathcal{R}$, an index $i \in 1..n$ and $o'_\infty \in O_\infty$ such that $o_\infty = (\prod_{k \in 1..i-1} o_{k-1} * O(j_k)) * o_{i-1} *^{\mathsf{m}} o'_\infty$, for all $k < i$,

$\mathcal{R} \vdash_\mu j_k$ and $\langle C(j_i), o'_\infty \rangle \in \mathcal{S}$. Then, the needed rule is $\mathsf{div}(\rho,\ i,\ o'_\infty)$.

- If $r_\infty \in R$, then, by construction of $\mathcal{S}'$, we have $\mathcal{R} \vdash_\mu c \Rightarrow \langle r_\infty, o_\infty \rangle$, hence there is a rule $\rho = \mathsf{rule}(j_1 \dots j_n,\ o_0 \dots o_n,\ c,\ r_\infty) \in \mathcal{R} \subseteq \mathcal{R}_\infty$, where, for all $k \in 1..n$, $\mathcal{R} \vdash_\mu j_k$ holds, and so $\langle C(j_k), R(j_k), O(j_k) \rangle \in \mathcal{S}'$.

□

We start by focusing on Theorem 6.40, the completeness result for the set of corules $C^{\mathcal{R}}_{\mathsf{eg}}$. First of all we have to characterise infinite computations by $\longrightarrow_{\mathcal{R}}$.

LEMMA 6.47 : If $\dfrac{}{c \Rightarrow ?} \xrightarrow{\sigma}{}^{\omega}_{\mathcal{R}}$, then $\sigma = u\sigma'$, for some $u \in O^\star$ and $\sigma' \in O^\omega$ such that

$$\frac{}{c \Rightarrow ?} \xrightarrow{u}{}^{\star}_{\mathcal{R}^{(\mathsf{pev}_?(\rho,\, i,\, ?))}} \frac{\tau_1 \ \dots \ \tau_{i-1}\ c_i \Rightarrow ?}{c \Rightarrow ?} \xrightarrow{\sigma'}{}^{\omega}_{\mathcal{R}} \text{ and } \frac{}{c_i \Rightarrow ?} \xrightarrow{\sigma'}{}^{\omega}_{\mathcal{R}}$$

Proof: Let $\sigma = (o_i)_{i \in \mathbb{N}}$, then there is an infinite sequence $(\tau_i)_{i \in \mathbb{N}}$ such that $\tau_0 = \dfrac{}{c \Rightarrow ?}$ and $\tau_i \xrightarrow{o_i}_{\mathcal{R}} \tau_{i+1}$, for all $i \in \mathbb{N}$. By definition of $\longrightarrow_{\mathcal{R}}$, for all $i \in \mathbb{N}$, since $\tau_i \xrightarrow{o_i}_{\mathcal{R}} \tau_{i+1}$, we have $\mathsf{r}(\tau_i) = c \Rightarrow ?$. For all $i \in \mathbb{N}$, let $\mathsf{br}(\tau_i)$ be the number of direct subtrees of τ_i, then, by definition of $\longrightarrow_{\mathcal{R}}$, $(\mathsf{br}(\tau_i))_{i \in \mathbb{N}}$ is an increasing sequence of natural numbers. By Assumption 6.1, such sequence is bounded, hence there is $n \in \mathbb{N}$ such that $\mathsf{br}(\tau_k) = \mathsf{br}(\tau_n)$, for all $k \geq n$. Let $n \in \mathbb{N}$ be the least number with such property, and note that $n \geq 1$ and $\mathsf{br}(\tau_n) \geq 1$, because $\mathsf{br}(\tau_1) = 1$, since the step $\tau_0 \xrightarrow{o_0}_{\mathcal{R}} \tau_1$ is done using (LTR-2); further, since n is the least index with such property, we have $\mathsf{br}(\tau_{n-1}) < \mathsf{br}(\tau_n)$, hence we have

$$\tau_n = {}_{(\mathsf{pev}_?(\rho,\, i,\, ?))} \frac{\tau'_1 \ \dots \ \tau'_{i-1}\ c_i \Rightarrow ?}{c \Rightarrow ?}$$

Then, for all $k \geq n$, the step $\tau_k \xrightarrow{o_k}_{\mathcal{R}} \tau_{k+1}$ is done applying (LTR-5), because (LTR-1) and (LTR-3) imply τ_{k+1} is complete, which is not possible as $\mathsf{r}(\tau_{k+1}) = c \Rightarrow ?$, and (LTR-2) and (LTR-4) imply $\mathsf{br}(\tau_{k+1}) > \mathsf{br}(\tau_k)$, which is not possible as $\mathsf{br}(\tau_{k+1}) = \mathsf{br}(\tau_k) = \mathsf{br}(\tau_n)$. Therefore, for all $k \geq n$, we have

$$\tau_k = {}_{(\mathsf{pev}_?(\rho,\, i,\, ?))} \frac{\tau'_1 \ \dots \ \tau'_{i-1}\ \tau'_{ik}}{c \Rightarrow ?}$$

and $\tau'_{ik} \xrightarrow{o_k}_{\mathcal{R}} \tau'_{i(k+1)}$ and $\tau'_{i0} = \dfrac{}{c_i \Rightarrow ?}$. Hence, if we set $u = o_0 \dots o_{n-1}$ and $\sigma' = (o_{n+k})_{k \in \mathbb{N}}$, then we have $\sigma = u\sigma'$, $\dfrac{}{c \Rightarrow ?} \xrightarrow{u}{}^{\star}_{\mathcal{R}} \tau_n \xrightarrow{\sigma'}{}^{\omega}_{\mathcal{R}}$ and $\dfrac{}{c_i \Rightarrow ?} \xrightarrow{\sigma'}{}^{\omega}_{\mathcal{R}}$, as needed. □

To check Item 1 of Lemma 6.46, we need the following results:

LEMMA 6.48 : If $\tau \xrightarrow{o}_{\mathcal{R}} \tau'$, with $o \neq \mathsf{e}$, and $C(\mathsf{r}(\tau)) = c$, then $\mathcal{R}_\infty \cup C^{\mathcal{R}}_{\mathsf{eg}} \vdash_\mu c \Rightarrow \langle \infty, O(\tau) * o *^{\mathsf{m}} o_\infty \rangle$, for all $o_\infty \in O_\infty$.

Proof: The proof is by induction on the definition of $\longrightarrow_{\mathcal{R}}$.

Case: (LTR-1),(LTR-3) We know that $\tau' = {}_{(\rho)}\dfrac{\tau_1 \ \ldots \ \tau_n}{c \Rightarrow \langle r, o' \rangle}$, with $\rho = \mathsf{rule}(j_1 \ldots j_n,\ o_0 \ldots o_n,\ c,\ r)$ in $\mathcal{R}$ and $o = o_n \neq \mathsf{e}$ and $\mathcal{R} \vdash_\mu j_i$, for all $i \in 1..n$. By co-gen$(\rho,\ n,\ o_\infty)$, which is in $C_{\mathsf{eg}}^{\mathcal{R}}$ as $o_n \neq \mathsf{e}$, applied to $j_1, \ldots, j_n$, we get $\mathcal{R}_\infty \cup C_{\mathsf{eg}}^{\mathcal{R}} \vdash_\mu c \Rightarrow \langle \infty, o' *^{\mathsf{m}} o_\infty \rangle$, but, by Lemma 6.25, we have $o' = O(\tau') = O(\tau) * o$, as needed.

Case: (LTR-2),(LTR-4) We know that $\tau' = {}_{(\mathsf{pev}_?(\rho,\, i+1,\, ?))}\dfrac{\tau_1 \ \ldots \ \tau_i \ c' \Rightarrow ?}{c \Rightarrow ?}$, with $\rho = \mathsf{rule}(j_1 \ldots j_n,\ o_0 \ldots o_n,\ c,\ r)$, $i \in 0..n-1$ and $o = o_i \neq \mathsf{e}$ and $\mathcal{R} \vdash_\mu j_k$, for all $k \in 1..i$. By co-gen$(\rho,\ i,\ o_\infty)$, which is in $C_{\mathsf{eg}}^{\mathcal{R}}$ as $o_i \neq \mathsf{e}$, applied to $j_1, \ldots, j_i$, we get $\mathcal{R}_\infty \cup C_{\mathsf{eg}}^{\mathcal{R}} \vdash_\mu c \Rightarrow \langle \infty, o' * o_i *^{\mathsf{m}} o_\infty \rangle$, with $o' = \prod_{k \in 1..i} o_{k-1} * O(j_k) = O(\tau)$, as needed.

Case: (LTR-5) We know that $\tau = {}_{(\mathsf{pev}_?(\rho,\, i,\, ?))}\dfrac{\tau_1 \ \ldots \ \tau_{i-1} \ \tau_i}{c \Rightarrow ?}$, $\tau' = {}_{(\mathsf{pev}_?(\rho,\, i,\, r_?))}\dfrac{\tau_1 \ \ldots \tau_{i-1} \ \tau_i'}{c \Rightarrow ?}$ and $\tau_i \xrightarrow{o}_{\mathcal{R}} \tau_i'$, with $\rho = \mathsf{rule}(j_1 \ldots j_n,\ o_0 \ldots o_n,\ c,\ r)$ and $i \in 1..n$ and $\mathcal{R} \vdash_\mu j_k$, for all $k \in 1..i-1$. Let $C(\tau_i) = C(\tau_i') = c_i$. By induction hypothesis, we get $\mathcal{R}_\infty \cup C_{\mathsf{eg}}^{\mathcal{R}} \vdash_\mu c_i \Rightarrow \langle \infty, O(\tau_i) * o *^{\mathsf{m}} o_\infty \rangle$, for all $o_\infty \in O_\infty$; then, applying $\mathsf{div}(\rho,\ i,\ O(\tau_i) * o *^{\mathsf{m}} o_\infty)$, which is in $\mathcal{R}_\infty$ by definition, to $j_1, \ldots, j_{i-1}$ and $c_i \Rightarrow \langle \infty, O(\tau_1) * o *^{\mathsf{m}} o_\infty \rangle$, we get $\mathcal{R}_\infty \cup C_{\mathsf{eg}}^{\mathcal{R}} \vdash_\mu c \Rightarrow \langle \infty, o' * O(\tau_i) * o *^{\mathsf{m}} o_\infty \rangle$, with $o' = (\prod_{k \in 1..i-1} o_{k-1} * O(j_k)) * o_{i-1}$. Since $O(j_k) = O(\tau_k)$, for all $k \in 1..i-1$, we get $o' * O(\tau_i) = O(\tau)$, as needed.

□

LEMMA 6.49 : If $\tau \xrightarrow{\sigma}{}^{\omega}_{\mathcal{R}}$, with $C(\mathsf{r}(\tau)) = c$, and $\sigma = o_0 \ldots o_k \sigma'$, with $o_k \neq \mathsf{e}$, then $\mathcal{R}_\infty \cup C_{\mathsf{eg}}^{\mathcal{R}} \vdash_\mu c \Rightarrow \langle \infty, O(\tau) * o_0 * \cdots * o_k *^{\mathsf{m}} \mathsf{it}^\infty(\sigma') \rangle$.

Proof: Let $\sigma = o_0 \ldots o_k \sigma'$ and $u = o_0 \ldots o_{k-1}$, then $\tau \xrightarrow{u}{}^{\star}_{\mathcal{R}} \tau_1 \xrightarrow{o_k}_{\mathcal{R}} \tau_2 \xrightarrow{\sigma'}{}^{\omega}_{\mathcal{R}}$ with $C(\mathsf{r}(\tau)) = C(\mathsf{r}(\tau_1)) = C(\mathsf{r}(\tau_2)) = c$. By Lemma 6.48, we have $\mathcal{R}_\infty \cup C_{\mathsf{eg}}^{\mathcal{R}} \vdash_\mu c \Rightarrow \langle \infty, O(\tau_1) * o_k *^{\mathsf{m}} \mathsf{it}^\infty(\sigma') \rangle$ and, by Lemma 6.26, we get $O(\tau_1) = O(\tau) * \mathsf{it}(v)$, hence we have $O(\tau_1) * o_k *^{\mathsf{m}} \mathsf{it}^\infty(\sigma') = O(\tau) * \mathsf{it}(v o_k) *^{\mathsf{m}} \mathsf{it}^\infty(\sigma') = O(\tau) *^{\mathsf{m}} \mathsf{it}^\infty(\sigma)$, as needed. □

To check Item 2 of Lemma 6.46, we rely on the following lemma:

LEMMA 6.50 : If $\dfrac{}{c \Rightarrow ?} \xrightarrow{\sigma}{}^{\omega}_{\mathcal{R}}$, then there is $\rho = \mathsf{rule}(j_1 \ldots j_n,\ o_0 \ldots o_n,\ c,\ r)$ in $\mathcal{R}$, an index $i \in 1..n$ and $\sigma' \in O^\omega$ such that $\sigma = v\sigma'$ for some $v \in O^\star$ and

- $\mathcal{R} \vdash_\mu j_k$, for all $k \in 1..i-1$,
- $\dfrac{}{C(j_i) \Rightarrow ?} \xrightarrow{\sigma'}{}^{\omega}_{\mathcal{R}}$, and
- $\mathsf{it}^\infty(\sigma) = (\prod_{k \in 1..i-1} o_{k-1} * O(j_k)) * o_{i-1} *^{\mathsf{m}} \mathsf{it}^\infty(\sigma')$.

Proof: Note that, by Lemma 6.47, we have $\sigma = \upsilon\sigma'$ and

$$\frac{}{c \Rightarrow ?} \xrightarrow{\upsilon}{}^{\star}_{\mathcal{R}} \tau = {}_{(\mathsf{pev}_?(\rho,i,?))}\frac{\tau_1 \ \ldots \ \tau_i}{c \Rightarrow ?} \xrightarrow{\sigma'}{}^{\omega}_{\mathcal{R}} \text{ and } \tau_i = \frac{}{c_i \Rightarrow ?} \xrightarrow{\sigma'}{}^{\omega}_{\mathcal{R}}$$

with $\rho = \mathsf{rule}(j_1 \ldots j_n,\ o_0 \ldots o_n,\ c,\ r)$ in $\mathcal{R}$ and $i \in 1..n$ and $\mathcal{R} \vdash_\mu j_k$, for all $k \in 1..i-1$, as τ_k is complete. To conclude, note that, by Lemma 6.26, we have $\mathsf{it}(\upsilon) = O(\tau) = (\prod_{k\in 1..i-1} o_{k-1} * O(\tau_k)) * o_{i-1}$, then, since $O(\tau_k) = O(j_k)$, for all $k \in 1..i-1$, we get

$$\mathsf{it}^\infty(\sigma) = \mathsf{it}^\infty(\upsilon\sigma') = \mathsf{it}(\upsilon) *^{\mathsf{m}} \mathsf{it}^\infty(\sigma') = (\prod_{k\in 1..i-1} o_{k-1} * O(j_k)) * o_{i-1} *^{\mathsf{m}} \mathsf{it}^\infty(\sigma')$$

□

We have now all the elements to prove completeness.

Proof (Theorem 6.40)*:* The proof is by Lemma 6.46. Let $\mathcal{S} \subseteq C \times O_\infty$ be the set $\{\langle c, o_\infty\rangle \in C \times O_\infty \mid c \rightsquigarrow_{\mathcal{R}} \langle \infty, o_\infty\rangle\}$ and consider $\langle c, o_\infty\rangle \in \mathcal{S}$, hence we know that $\frac{}{c \Rightarrow ?} \xrightarrow{\sigma}{}^{\omega}_{\mathcal{R}}$ and $\mathsf{it}^\infty(\sigma) = o_\infty$, for some $\sigma \in O^\omega$.

To check Item 1 of Lemma 6.46, we have to prove that $\mathcal{R}_\infty \cup C^{\mathcal{R}}_{\mathsf{eg}} \vdash_\mu c \Rightarrow \langle \infty, o_\infty\rangle$ holds. We have two cases: if $\sigma = \mathsf{e}^\omega$, then we get the thesis by (CO-UNIT); if $\sigma = o_0 \ldots o_k\sigma'$, with $o_k \neq \mathsf{e}$, then the thesis follows from Lemma 6.49.

Item 2 of Lemma 6.46 follows immediately by Lemma 6.50. □

We now prove our second completeness result, concerning the set of corules $C^{\mathcal{R}}_{\mathsf{e}}$. To show this result, we rely on some key properties of finitely generated observations (cf. Lemma 6.14) and on the next lemma, relating the transition relation to the set of elementary observations $E_{\mathcal{R}}$.

LEMMA 6.51 : If $\tau \xrightarrow{o}_{\mathcal{R}} \tau'$, then $o \in E_{\mathcal{R}}$.

Proof: The proof is by induction on the definition of $\xrightarrow{}_{\mathcal{R}}$. For clauses (LTR-1), (LTR-2), (LTR-3) and (LTR-4) the thesis is immediate as we have $o = E(\rho, i)$ for some $\rho \in \mathcal{R}$ and $i \in 1..\#\rho$. For clause (LTR-5), the thesis follows by induction hypothesis. □

LEMMA 6.52 : If $\frac{}{c \Rightarrow ?} \xrightarrow{\sigma}{}^{\omega}_{\mathcal{R}}$, with $\sigma = (o_i)_{i\in\mathbb{N}}$ and $u_i = o_0 \ldots o_{i-1}$, for all $i \in \mathbb{N}$, then, for all $i \in \mathbb{N}$, there is $k \geq i$ such that $\mathcal{R}_\infty \cup C^{\mathcal{R}}_{\mathsf{e}} \vdash_\mu c \Rightarrow \langle \infty, \mathsf{it}^\infty(u_k\mathsf{e}^\omega)\rangle$.

Proof: By Lemma 6.47, we have $\sigma = \upsilon\sigma'$, with $\upsilon \neq \varepsilon$, and

$$\frac{}{c \Rightarrow ?} \xrightarrow{\upsilon}{}^{\star}_{\mathcal{R}} \tau' = {}_{(\mathsf{pev}_?(\rho,n,?))}\frac{\tau_1 \ \ldots \ \tau_n}{c \Rightarrow ?} \xrightarrow{\sigma'}{}^{\omega}_{\mathcal{R}} \text{ and } \tau_n = \frac{}{c_n \Rightarrow ?} \xrightarrow{\sigma'}{}^{\omega}_{\mathcal{R}}$$

with $\rho = \mathsf{rule}(j_1 \ldots j_m,\ o_0 \ldots o_m,\ c,\ r)$ in $\mathcal{R}$ and $n \in 1..m$. The proof is by complete arithmetic induction on the length of u_i, namely on $i \in \mathbb{N}$. If u_i is a

prefix of v, then, by (CO-UNIT), we get $\mathcal{R}_\infty \cup C_{\mathsf{e}}^{\mathcal{R}} \vdash_\mu c_n \Rightarrow \langle \infty, \mathsf{it}^\infty(u'\mathsf{e}^\omega)\rangle$, with $u' = \varepsilon$. Otherwise, $u_i \neq v$ and v is a prefix of u_i, say $u_i = vu$, then $\sigma' = u\sigma''$ and u is strictly shorter than u_i, as v is not empty. Therefore, by induction hypothesis, we get $\mathcal{R}_\infty \cup C_{\mathsf{e}}^{\mathcal{R}} \vdash_\mu c_n \Rightarrow \langle \infty, \mathsf{it}^\infty(u'\mathsf{e}^\omega)\rangle$, where u is a prefix of u', hence u_i is a prefix of vu'.

In both cases we have $\mathcal{R}_\infty \cup C_{\mathsf{e}}^{\mathcal{R}} \vdash_\mu c_n \Rightarrow \langle \infty, \mathsf{it}^\infty(u'\mathsf{e}^\omega)\rangle$, with u_i a prefix of vu'. Let $o_\infty = \mathsf{it}^\infty(u'\mathsf{e}^\omega)$, then, applying $\mathsf{div}(\rho,\ n,\ o_\infty)$ to $j_1, \ldots, j_{n-1}$ and $c_n \Rightarrow \langle \infty, o_\infty\rangle$, we get $\mathcal{R}_\infty \cup C_{\mathsf{e}}^{\mathcal{R}} \vdash_\mu c \Rightarrow \langle \infty, o'_\infty\rangle$, with $o'_\infty = (\prod_{i\in 1..n-1} o_{i-1} * O(j_i)) * o_{i-1} *^{\mathsf{m}} o_\infty$. Since, for all $i \in 1..n-1$, we have $O(j_i) = O(\tau_i)$, and $O(\tau_n) = \mathsf{e}$, we get $o'_\infty = O(\tau') *^{\mathsf{m}} \mathsf{it}^\infty(u'\mathsf{e}^\omega)$; then, by Lemma 6.26, as $\tau \xrightarrow{v} {}^\star_{\mathcal{R}}\, \tau'$, we have $O(\tau') = \mathsf{it}(v)$, thus $o'_\infty = \mathsf{it}^\infty(vu'\mathsf{e}^\omega)$, which proves the thesis, as u_i is a prefix of vu'. □

The proof of Theorem 6.41 is essentially the same as the one of Theorem 6.40: we apply Lemma 6.46, where Lemmas 6.14 and 6.49 assure Item 1, and Lemma 6.50 assures Item 2.

Proof (Theorem 6.41)*:* The proof is by Lemma 6.46. Let $\mathcal{S} \subseteq C \times O_\infty$ be the set $\{\langle c, o_\infty\rangle \in C \times O_\infty \mid c \rightsquigarrow_{\mathcal{R}} \langle \infty, o_\infty\rangle$ and o_∞ is finitely generated$\}$ and consider $\langle c, o_\infty\rangle \in \mathcal{S}$, hence we know that $\dfrac{}{c \Rightarrow ?} \xrightarrow{\sigma}{}^\omega_{\mathcal{R}}$ and $\mathsf{it}^\infty(\sigma) = o_\infty$ is finitely generated, for some $\sigma = (o_i)_{i\in\mathbb{N}} \in O^\omega$.

To check Item 1 of Lemma 6.46, we have to prove that $\mathcal{R}_\infty \cup C_{\mathsf{e}}^{\mathcal{R}} \vdash_\mu c \Rightarrow \langle \infty, o_\infty\rangle$ holds. Let $u_i = o_0 \ldots o_i$, for all $i \in \mathbb{N}$. By Lemma 6.51, $o_i \in E_{\mathcal{R}}$ for all $i \in \mathbb{N}$, hence, by Lemma 6.14 (1), there is $n \in \mathbb{N}$ such that, for all $k \geq n$, $o_\infty = \mathsf{p}(u_k\mathsf{e}^\omega) = \mathsf{it}^\infty(u_k\mathsf{e}^\omega)$. By Lemma 6.52, there is $h \geq n$ such that $\mathcal{R}_\infty \cup C_{\mathsf{e}}^{\mathcal{R}} \vdash_\mu c \Rightarrow \langle \infty, \mathsf{it}^\infty(u_h\mathsf{e}^\omega)\rangle$, hence we get the thesis as $o_\infty = \mathsf{it}^\infty(u_h\mathsf{e}^\omega)$.

Item 2 of Lemma 6.46 follows immediately by Lemma 6.50, because $\sigma = v\sigma'$ and $\mathsf{p}(\sigma')$ is finitely generated by Lemma 6.14 (2). □

SOUNDNESS First of all, we construct computations in $\longrightarrow_{\mathcal{R}}$ starting from infinite evaluation trees in $\mathcal{R}_\infty$. More precisely, we show that, for any infinite evaluation tree τ in $\mathcal{R}_\infty$ of $c \Rightarrow \langle \infty, o_\infty\rangle$, there is an infinite reduction $\dfrac{}{c \Rightarrow ?} \xrightarrow{\sigma}{}^\omega_{\mathcal{R}}$ where σ is equivalent to σ_τ (see the definition introduced for Lemma 6.43), as stated in the following lemma.

LEMMA 6.53 : If τ is an infinite evaluation tree in $\mathcal{R}_\infty$ of $c \Rightarrow \langle \infty, o_\infty\rangle$, then $\dfrac{}{c \Rightarrow ?} \xrightarrow{\sigma}{}^\omega_{\mathcal{R}}$, for some $\sigma \in O^\omega$ such that $\mathsf{it}^\infty(\sigma) = \mathsf{it}^\infty(\sigma_\tau)$.

Proof: The infinite evaluation tree τ can be described by a sequence $(\tau_i)_{i\in\mathbb{N}}$ defined as follows:

$$\tau_0 = \tau \qquad \tau_i = {}_{(\mathsf{div}(\rho_i,\ k_i,\ o_\infty^{i+1}))}\frac{\tau_{i,1}\ \ldots\ \tau_{i,k_i-1}\ \tau_{i+1}}{c_i \Rightarrow \langle \infty,\ o_\infty^i\rangle}$$

where, for all $i \in \mathbb{N}$, $\rho_i = \mathsf{rule}(j_{i,1} \ldots j_{i,n_i},\ o_{i,0} \ldots o_{i,n_i},\ c_i,\ r_i)$ is a rule in $\mathcal{R}$, $k_i \in 1..n_i$ and $o_\infty^i \in O_\infty$ and $o_\infty^0 = o_\infty$. We define for all $i \in \mathbb{N}$ and $h \in 0..i$ a

finite partial evaluation tree $\tau^\star_{i,h}$ as follows:

$$\tau^\star_{i,i} = {}_{(\mathsf{ax}_?(c_i))}\frac{}{c_i \Longrightarrow ?} \qquad (h < i)\quad \tau^\star_{i,h} = {}_{(\mathsf{div}(\rho_h,\, k_h,\, o_\infty^{h+1}))}\frac{\tau_{h,1}\ \ldots\ \tau_{h,k_h-1}\ \tau^\star_{i,h+1}}{c_h \Longrightarrow ?}$$

We prove that, for all $i \in \mathbb{N}$ and $h \in 0..i$, $\tau^\star_{i,h} \xrightarrow{u_i}{}^\star_{\mathcal{R}}\, \tau^\star_{i+1,h}$, for some $u_i \in O^+$ with $\mathsf{it}(u_i) = \sigma_\tau(i)$. The proof is by induction on $i - h$. If $i - h = 0$, that is, $h = i$, then, by Lemma 6.27, we get $\tau^\star_{i,i} \xrightarrow{u_i}{}^\star_{\mathcal{R}}\, \tau^\star_{i+1,i}$, with $\mathsf{it}(u_i) = O(\tau^\star_{i+1,i})$. By definition we have $O(\tau_{i+1,i}) = o_{i,0} * (\prod_{k\in 1..k_i-1} O(j_{i,k}) * o_{i,k}) = \sigma_\tau(i)$. If $i - h = k + 1$, then we have

$$\tau^\star_{i,h} = {}_{(\mathsf{div}(\rho_h,\, k_h,\, o_\infty^{h+1}))}\frac{\tau_{h,1}\ \ldots\ \tau_{h,k_h-1}\ \tau^\star_{i,h+1}}{c_h \Longrightarrow ?}$$

since $i-(h+1) = k$, by induction hypothesis, we get $\tau^\star_{i,h1} \xrightarrow{u_i}{}^\star_{\mathcal{R}}\, \tau^\star_{i+1,h+1}$ with $\mathsf{it}(u_i) = \sigma_\tau(i)$, hence, applying (LTR-5), we get $\tau^\star_{i,h} \xrightarrow{u_i}{}^\star_{\mathcal{R}}\, \tau^\star_{i+1,h}$, as needed.

Let σ be the flattening of $(u_i)_{i\in\mathbb{N}}$, that is, $\sigma = u_0u_1u_2\ldots$, from what we have just proved we have $\tau_{0,0} = \dfrac{}{c \Longrightarrow ?} \xrightarrow{\sigma}{}^\omega_{\mathcal{R}}$ and since, for all $i \in \mathbb{N}$, $\mathsf{it}(u_i) = \sigma_\tau(i)$, we have $\sigma \bowtie \sigma_\tau$, and so $\mathsf{it}^\infty(\sigma) = \mathsf{p}(\sigma) = \mathsf{p}(\sigma_\tau) = \mathsf{it}^\infty(\sigma_\tau)$, by the infinite associative law. □

To prove soundness of $C^{\mathcal{R}}_{\mathsf{e}}$ (cf. Theorem 6.42), we need some properties of observations derivable in $\mathcal{R}_\infty \cup C^{\mathcal{R}}_{\mathsf{e}}$.

LEMMA 6.54 : If $\mathcal{R}_\infty \cup C^{\mathcal{R}}_{\mathsf{e}} \vdash_\mu c \Longrightarrow \langle \infty, o_\infty\rangle$, then

1. if $\mathcal{R} \vdash_\mu c \Longrightarrow \langle r, o\rangle$, then $o_\infty \in O$ and $o_\infty \preceq_* o$,
2. if τ is an infinite evaluation tree in $\mathcal{R}_\infty$ of $c \Longrightarrow \langle \infty, o'_\infty\rangle$, then $o_\infty = \mathsf{it}^\infty(\sigma_\tau[h]\cdot o \cdot \mathsf{e}^\omega)$, for some $h \in \mathbb{N}$ and $o \in O$ such that $o \preceq_* \sigma_\tau(h)$.

Proof: We prove both items by induction on rules in $\mathcal{R}_\infty \cup C^{\mathcal{R}}_{\mathsf{e}}$. There are only two relevant cases.

Case: co-unit(c) We have $o_\infty = \mathsf{e}$, hence we get Item 1, as $\mathsf{e} \preceq_* o$, and Item 2, as $\mathsf{e} = \mathsf{it}^\infty(\mathsf{e}^\omega) = \mathsf{it}^\infty(\sigma_\tau[0]\cdot \mathsf{e}\cdot \mathsf{e}^\omega)$.

Case: $\mathsf{div}(\rho,\, i,\, \hat{o_\infty})$ We have $\rho = \mathsf{rule}(j_1\ldots j_n,\, o_0\ldots o_n,\, c,\, \hat{r})$, $i \in 1..n$, $o_\infty = o_0 * (\prod_{k\in 1..i-1} O(j_k) * o_k) *^{\mathsf{m}} \hat{o_\infty}$, and $\mathcal{R}_\infty \cup C^{\mathcal{R}}_{\mathsf{e}} \vdash_\mu j_k$, for all $k \in 1..i-1$, and $\mathcal{R}_\infty \cup C^{\mathcal{R}}_{\mathsf{e}} \vdash_\mu C(j_i) \Longrightarrow \langle \infty, \hat{o_\infty}\rangle$. By Theorem 6.31, we get $\mathcal{R} \vdash_\mu j_k$, for all $k \in 1..i-1$. We prove separately the two items:

1. Suppose $c \Longrightarrow \langle r, o\rangle$ is derived by $\rho' = \mathsf{rule}(j'_1\ldots j'_m,\, o'_0\ldots o'_m,\, c,\, r)$, then, since $\mathcal{R} \vdash_\mu j'_k$, for all $k \in 1..m$, using Lemma 6.37 and Assumption 6.2 we can prove $i \le m$, $o_0 = o'_0$, $o_k = o'_k$ and $j_k = j'_k$, for all $k \in 1..i-1$ and $C(j_i) = C(j'_i)$. We also know that $\mathcal{R} \vdash_\mu j'_i$, hence, by induction hypothesis, we get $\hat{o_\infty} \in O$ and $\hat{o_\infty} \preceq_* O(j'_i)$. Therefore, $o_\infty = o'_0 * (\prod_{k\in 1..i-1} O(j'_k) * o'_k) * \hat{o_\infty} \preceq_* o'_0 * (\prod_{k\in 1..m} O(j'_k) * o'_k) = o$, as needed.

2. Suppose $c \Rightarrow \langle \infty, o'_\infty \rangle$ has an infinite evaluation tree τ in $\mathcal{R}_\infty$, then

$$\tau = {}_{(\mathsf{div}(\rho', l, o''_\infty))} \frac{\tau_1 \ \ldots \ \tau_{l-1} \ \tau_l}{c \Rightarrow \langle \infty, o_\infty \rangle}$$

where $\rho' = \mathsf{rule}(j'_1 \ldots j'_m, o'_0 \ldots o'_m, c, r)$, $l \in 1..m$ and $o'_\infty = o'_0 * (\prod_{k \in 1..l-1} O(j'_k) * o'_k) *^{\mathsf{m}} o''_\infty$. Since $\mathsf{r}(\tau_k) = j'_k$, for all $k \in 1..l-1$, by Theorem 6.31 we have $\mathcal{R} \vdash_\mu j'_k$, for all $k \in 1..l-1$. Again, using Lemma 6.37 and Assumption 6.2, setting $h = \min\{i, l\}$, we get $o_0 = o'_0$, $o_k = o'_k$ and $j_k = j'_k$, for all $k \in 1..h-1$, and $C(j_h) = C(j'_h)$. We have three cases.

- If $l < i$, then, since τ_l is an infinite evaluation tree of $C(j'_l) \Rightarrow \langle \infty, o''_\infty \rangle$, we have $\mathcal{R}_\infty \vdash_\nu C(j'_l) \Rightarrow \langle \infty, o''_\infty \rangle$ and, since $\mathcal{R} \vdash_\mu j_l$, by Lemma 6.38, we get a contradiction.
- If $i < l$, then, by Item 1, we have $\hat{o_\infty} \in O$ and $\hat{o_\infty} \preceq_* O(j'_i)$, hence $o_\infty = o'_0 * (\prod_{k \in 1..i-1} O(j'_k) * o'_k) *^{\mathsf{m}} \hat{o_\infty} \in O$ and $o_\infty \preceq_* o'_0 * (\prod_{k \in 1..l-1} O(j'_k) * o'_k) = \sigma_\tau(0)$, because $i < l$. Therefore, we get $o_\infty = \mathsf{it}^\infty(\varepsilon \cdot o_\infty \cdot \mathsf{e}^\omega) = \mathsf{it}^\infty(\sigma_\tau[0] \cdot o_\infty \cdot \mathsf{e}^\omega)$, with $o_\infty \preceq_* \sigma_\tau(0)$, as needed.
- If $i = l$, then, since τ_l is an infinite evaluation tree of $C(j'_l) \Rightarrow \langle \infty, o''_\infty \rangle$, by induction hypothesis we get $\hat{o_\infty} = \mathsf{it}^\infty(\sigma_{\tau_l}[h] \cdot o \cdot \mathsf{e}^\omega)$ for some $h \in \mathbb{N}$ such that $o \preceq_* \sigma_{\tau_l}(h)$. Since $o_\infty = o'_0 * (\prod_{k \in 1..i-1} O(j'_k) * o'_k) *^{\mathsf{m}} \hat{o_\infty}$ and $\sigma_\tau(0) = o'_0 * (\prod_{k \in 1..i-1} O(j'_k) * o'_k)$, we get $o_\infty = \mathsf{it}^\infty(\sigma_\tau(0) \cdot \sigma_{\tau_l}[h] \cdot o \cdot \mathsf{e}^\omega)$ with $o \preceq_* \sigma_{\tau_l}(h)$. By definition of σ_τ, we have, for all $i \in \mathbb{N}$, $\sigma_{\tau_l}(i) = \sigma_\tau(i+1)$, hence $\sigma_\tau[i+1] = \sigma_\tau(0) \cdot \sigma_{\tau_l}[i]$; therefore we get $o_\infty = \mathsf{it}^\infty(\sigma_\tau[h+1] \cdot o \cdot \mathsf{e}^\omega)$ and $o \preceq_* \sigma_\tau(h+1)$, as needed.

□

We can now prove the soundness result for $C_{\mathsf{e}}^{\mathcal{R}}$ (Theorem 6.42).

Proof (Theorem 6.42)*:* By hypothesis, we know that $c \Rightarrow \langle \infty, o_\infty \rangle$ has an infinite evaluation tree τ in $\mathcal{R}_\infty$, and $\mathcal{R}_\infty \cup C_{\mathsf{e}}^{\mathcal{R}} \vdash_\mu c \Rightarrow \langle \infty, o_\infty \rangle$ holds. By Lemma 6.53, we have $\dfrac{}{c \Rightarrow ?} \xrightarrow{\sigma}{}^{\omega}_{\mathcal{R}}$ and $\mathsf{it}^\infty(\sigma) = \mathsf{it}^\infty(\sigma_\tau)$, for some $\sigma \in O^\omega$, that is, $[\sigma]_\equiv = [\sigma_\tau]_\equiv$. By Lemma 6.54 (2), we also know that $o_\infty = \mathsf{it}^\infty(v\mathsf{e}^\omega)$, with $v = \sigma_\tau[h] \cdot o$ and $o \preceq_* \sigma_\tau(h)$, for some $h \in \mathbb{N}$. Therefore, in order to get the thesis, we just have to show that $\sigma_\tau \equiv v\mathsf{e}^\omega$.

We have $v\mathsf{e}\omega \sqsubseteq \sigma_\tau$, since $\mathsf{it}(v) = \mathsf{it}(\sigma_\tau[h]) * o \preceq_* \mathsf{it}(\sigma_\tau[h]) * \sigma_\tau(h) = \mathsf{it}(\sigma_{\tau[h+1]})$. On the other hand, recall that o_∞ is a limit product of σ_τ, hence, for all $n \in \mathbb{N}$, we have $o_\infty = \mathsf{it}(\sigma_\tau[n]) *^{\mathsf{m}} o'_\infty = \mathsf{it}(\sigma_\tau[n]) *^{\mathsf{m}} \mathsf{it}^\infty(\sigma') = \mathsf{it}^\infty(\sigma_\tau[n] \cdot \sigma')$. Hence, since $\mathsf{it}^\infty(v\mathsf{e}^\omega) = o_\infty = \mathsf{it}^\infty(\sigma_\tau[n] \cdot \sigma')$, we have $v\mathsf{e}^\omega \equiv \sigma_\tau[n] \cdot \sigma'$, for all $n \in \mathbb{N}$. Therefore, by definition of $\equiv$, there is a prefix v' of $v\mathsf{e}^\omega$ such that $\mathsf{it}(\sigma_\tau[n]) \preceq_* \mathsf{it}(v')$ and, since v' is a prefix of $v\mathsf{e}^\omega$, $\mathsf{it}(v') \preceq_* \mathsf{it}(v)$. Hence, for all $n \in \mathbb{N}$, $\mathsf{it}(\sigma_\tau[n]) \preceq_* \mathsf{it}(v)$, thus the least upper bound of $(\mathsf{it}(\sigma_\tau[n]))_{n \in \mathbb{N}}$, if any, is below $\mathsf{it}(v)$, and this shows $\sigma_\tau \sqsubseteq v\mathsf{e}^\omega$, proving the thesis. □

We now prove the two results concerning the set of corules $C^{\mathcal{R}}_{\mathsf{eg}}$.

LEMMA 6.55 : If τ is a finite partial evaluation tree with $C(\mathsf{r}(\tau)) = c$ and $O(\tau) \neq \mathsf{e}$, then $\mathcal{R}_\infty \cup C^{\mathcal{R}}_{\mathsf{eg}} \vdash_\mu c \Rightarrow \langle \infty, O(\tau) *^{\mathsf{m}} o_\infty \rangle$, for all $o_\infty \in O_\infty$.

Proof: Let us assume $\tau = {}_{(\rho_?)}\dfrac{\tau_1 \ \ldots \ \tau_i}{c \Rightarrow ?}$, where either $\rho_? = \rho \in \mathcal{R}$ or $\rho_? = \mathsf{pev}_?(\rho, i, R^O_?(\mathsf{r}(\tau_i)))$ and $\rho = \mathsf{rule}(j_1 \ldots j_n,\ o_0 \ldots o_n,\ c,\ r)$ and $i \in 0..n$. We know that either $o_k \neq \mathsf{e}$, for some $k \in 0..i$, or $O(\tau_k) \neq \mathsf{e}$, for some $k \in 1..i$, because otherwise we would have $O(\tau) = \mathsf{e}$, which is not possible. We distinguish these two cases. If $o_k \neq \mathsf{e}$ for some $k \in 0..i$, then, since we have $O(\tau) = o_0 * (\prod_{h \in 1..k} O(\tau_h) * o_h) * o$, for some $o \in O$, applying co-gen$(\rho,\ k,\ o *^{\mathsf{m}} o_\infty)$, which is defined as $o_k \neq \mathsf{e}$, we get $\mathcal{R}_\infty \cup C^{\mathcal{R}}_{\mathsf{eg}} \vdash_\mu c \Rightarrow \langle \infty, O(\tau) *^{\mathsf{m}} o_\infty \rangle$, as needed. If $O(\tau_k) \neq \mathsf{e}$, for some $k \in 1..i$, then, since by definition we have $O(\tau) = (\prod_{h \in 1..k} o_{h-1} * O(\tau_h)) * o$, for some $o \in O$, by induction hypothesis we get $\mathcal{R}_\infty \cup C^{\mathcal{R}}_{\mathsf{eg}} \vdash_\mu C(j_k) \Rightarrow \langle \infty, O(\tau_k) *^{\mathsf{m}} (o *^{\mathsf{m}} o_\infty) \rangle$, hence we get the thesis applying rule $\mathsf{div}(\rho,\ k,\ O(\tau_k) *^{\mathsf{m}} (o *^{\mathsf{m}} o_\infty))$. □

Proof (Lemma 6.43)*:* By hypothesis there is an infinite evaluation tree τ in $\mathcal{R}_\infty$ of $c \Rightarrow \langle \infty, o_\infty \rangle$, which can be described by a sequence $(\tau_i)_{i \in \mathbb{N}}$ defined as follows:

$$\tau_0 = \tau \qquad \tau_i = {}_{(\mathsf{div}(\rho_i,\ k_i,\ o^{i+1}_\infty))}\frac{\tau_{i,1} \ \ldots \tau_{i,k_i-1} \ \tau_{i+1}}{c_i \Rightarrow \langle \infty, o^i_\infty \rangle}$$

where, for all $i \in \mathbb{N}$, $\rho_i = \mathsf{rule}(j_{i,1} \ldots j_{i,n_i},\ o_{i,0} \ldots o_{i,n_i},\ c_i,\ r_i)$ is a rule in $\mathcal{R}$, $k_i \in 1..n_i$, $o^i_\infty \in O_\infty$, $o^0_\infty = o_\infty$ and $o^i_\infty = \sigma_\tau(i) *^{\mathsf{m}} o^{i+1}_\infty$, for all $i \in \mathbb{N}$. Furthermore, for all $i \in \mathbb{N}$, we have, by Theorem 6.31, $\mathcal{R} \vdash_\mu j_{i,k}$, for all $k \in 1..k_i - 1$, hence we can assume $\tau_{i,k}$ to be finite, and $\mathcal{R}_\infty \cup C^{\mathcal{R}}_{\mathsf{eg}} \vdash_\mu c_i \Rightarrow \langle \infty, o^i_\infty \rangle$.

The proof is by Lemma 6.46. Define the set $\mathcal{S} \subseteq C \times O_\infty$ as follows: $\langle \hat{c}, \hat{o}_\infty \rangle \in \mathcal{S}$ iff $\hat{c} = c_i$ and $\hat{o}_\infty$ is a limit product of σ_{τ_i}, for some $i \in \mathbb{N}$. Consider $\langle c_i, \hat{o}_\infty \rangle \in \mathcal{S}$, hence we know that $\langle \mathcal{R}_\infty, C^{\mathcal{R}}_{\mathsf{eg}} \rangle \vdash_\nu c_i \Rightarrow \langle \infty, o^i_\infty \rangle$, as $\mathsf{r}(\tau_i) = c_i \Rightarrow \langle \infty, o^i_\infty \rangle$ and $\hat{o}_\infty$ is a limit product of σ_{τ_i}, that is, there is a sequence $(\hat{o}^n_\infty)_{n \in \mathbb{N}}$ such that $\hat{o}_\infty = \hat{o}^0_\infty$ and, for all $n \in \mathbb{N}$, $\hat{o}^n_\infty = \sigma_{\tau_i}(n) *^{\mathsf{m}} \hat{o}^{n+1}_\infty$.

To check Item 1 of Lemma 6.46, we have to show that $\mathcal{R}_\infty \cup C^{\mathcal{R}}_{\mathsf{eg}} \vdash_\mu c_i \Rightarrow \langle \infty, \hat{o}_\infty \rangle$ holds. Since σ_τ is non-trivial, then σ_{τ_i} is non-trivial as well, hence there is $l \in \mathbb{N}$ such that $\sigma_{\tau_i}(l) \neq \mathsf{e}$. We prove that, for all $k \in 0..l$, $\mathcal{R}_\infty \cup C^{\mathcal{R}}_{\mathsf{eg}} \vdash_\mu c_{i+k} \Rightarrow \langle \infty, \hat{o}^k_\infty \rangle$ holds, which implies Item 1 as $\hat{o}_\infty = \hat{o}^0_\infty$. The proof is by induction on $l - k$. If $l - k = 0$, that is, $k = l$, then consider

$$\tau'_{i+l} = {}_{(\mathsf{pev}_?(\rho_{i+l}, k_{i+l}, R^O_?(j_{i+l,k_{i+l}-1}))}\frac{\tau_{i+l,1} \ \ldots \ \tau_{i+l,k_{i+l}-1}}{c_{i+l} \Rightarrow ?}$$

which is a finite partial evaluation tree and $O(\tau'_{i+l}) = \sigma_{\tau_{i+l}}(0) = \sigma_{\tau_i}(l) \neq \mathsf{e}$. Since $\hat{o}^l_\infty = \sigma_{\tau_i}(l) *^{\mathsf{m}} \hat{o}^{l+1}_\infty = O(\tau'_{i+l}) *^{\mathsf{m}} \hat{o}^{l+1}_\infty$, by Lemma 6.55 we get $\mathcal{R}_\infty \cup C^{\mathcal{R}}_{\mathsf{eg}} \vdash_\mu c_{i+l} \Rightarrow \langle \infty, o^l_\infty \rangle$, as needed. If $l - k = h + 1$, then, by induction hypothesis, we have that $\mathcal{R}_\infty \cup C^{\mathcal{R}}_{\mathsf{eg}} \vdash_\mu c_{i+k+1} \Rightarrow \langle \infty, \hat{o}^{k+1}_\infty \rangle$ holds. Since $\hat{o}^k_\infty = \sigma_{\tau_i}(k) *^{\mathsf{m}} \hat{o}^{k+1}_\infty$, we get the thesis by applying $\mathsf{div}(\rho_{i+k},\ k_{i+k},\ \hat{o}^{k+1}_\infty)$.

Item 2 of Lemma 6.46 immediately holds considering the rule ρ_i and index

$k_i \in 1..n_i$, because $\hat{o}_\infty = \sigma_{\tau_i}(0)\hat{o}^1_\infty = o_{i,0} * (\prod_{h \in 1..k_i-1} O(j_{i,h}) * o_{i,h}) *^{\mathsf{m}} \hat{o}^1_\infty$, and $\langle c_{i+1}, \hat{o}^1_\infty \rangle \in \mathcal{S}$, as $\hat{o}^1_\infty$ is a limit product of $\sigma_{\tau_{i+1}}$. □

Proof (Proposition 6.44)*:* Since $E_\mathcal{R} \subseteq O_\mathcal{R}$, the left-to-right implication is trivial. To prove the other direction, consider $\sigma \in O^\omega_\mathcal{R}$. We first show that there exists $\sigma' \in E^\omega_\mathcal{R}$ such that $\sigma \bowtie \sigma'$. Assume $\sigma = (o_i)_{i \in \mathbb{N}}$, then, by definition of $O_\mathcal{R}$, we have that $o_i = o_{i,1} * \cdots * o_{i,n_i}$ with $o_{i,1}, \ldots, o_{i,n_i} \in E_\mathcal{R}$, hence, defining σ' as the flattening of the sequence $(o_{i,1} \ldots o_{i,n_i})_{i \in \mathbb{N}}$, we have $\sigma \bowtie \sigma'$. Since σ is non-trivial, then σ' is non-trivial as well, because, if $o_i \neq \mathsf{e}$, then there is $k \in 1..n_i$ such that $o_{i,k} \neq \mathsf{e}$. Then, it is easy to check that o_∞ is a limit product of σ iff it is a limit product of σ', and this implies the thesis. □

Proof (Theorem 6.45)*:* Let τ be the infinite evaluation tree of $c \Rightarrow \langle \infty, o_\infty \rangle$ and $\sigma_\tau = (o_i)_{i \in \mathbb{N}}$. We have two cases:

- if σ_τ is trivial, that is, for all $k \geq n$, $o_k = \mathsf{e}$, for some $n \in \mathbb{N}$, then we can prove by induction on n that $\mathcal{R}_\infty \vdash \mathcal{C}^\mathcal{R}_\mathsf{e} c \Rightarrow \langle \infty, o_\infty \rangle$ holds (intuitively, because we can cut τ at depth n using (CO-UNIT)), hence by Theorem 6.42 we get the thesis;
- if σ_τ is non-trivial, then, since $\mathsf{it}^\infty(\sigma_\tau)$ and o_∞ are both limit products of σ_τ and $E_\mathcal{R}$ has unique limits, hence $O_\mathcal{R}$ has unique limits as well by Proposition 6.44, we have $o_\infty = \mathsf{it}^\infty(\sigma_\tau)$. By Lemma 6.53 we know there is $\sigma \in O^\omega$ such that $\dfrac{}{c \Rightarrow ?} \xrightarrow{\sigma}{}^\omega_\mathcal{R}$ and $\mathsf{it}^\infty(\sigma) = \mathsf{it}^\infty(\sigma_\tau)$, hence we get $c \rightsquigarrow_\mathcal{R} \langle \infty, o_\infty \rangle$, as needed.

□

7

Discussion

The big-step style can be useful for abstracting details, directly deriving the implementation of an interpreter, formally verifying compilers (Leroy and Grall, 2009), cost semantics, and soundness and completeness proofs for program logics (Charguéraud, 2013). However, modeling divergence is a non-trivial problem, even more when a non-terminating program can have a significant behaviour through observations. Indeed, standard, inductive, big-step semantics is able only to capture finite computation, hence it cannot distinguish between stuck and infinite computations.

In this part we address this problem, providing a systematic analysis of big-step semantics from an operational perspective. The first, and fundamental, methodological feature of our analysis is that we want to be independent from particular languages, developing an abstract study of big-step semantics in itself. Therefore, we provide a definition of what is, for us, a big-step semantics with or without observations and then our results will be applicable, as we show by several examples, to all concrete big-step semantics matching our definition.

A second important building block of our approach is that we take seriously the fact that big-step rules implicitly define an evaluation algorithm driven by rules. Indeed, we make such intuition formal by showing that using rules we can define a transition relation on incomplete derivations, abstractly modeling such evaluation algorithm. In presence of observations, this transition relation is labelled by observations produced by single transition steps, modeling their observable effect. Relying on this transition relation, we are able to define computations in the big-step semantics in the usual way, as possibly infinite sequences of transition steps; thus we can distinguish converging, diverging and stuck computation, even if big-step rules only define convergence. This shows that diverging and stuck computations are, in a sense, implicit in standard big-step rules, and the transition relation makes them explicit.

Finally, the third feature of our approach is that we provide constructions that, starting from a usual big-step semantics, produce an extended one where the distinction between diverging and stuck computation is explicit. Such constructions show that we can distinguish stuck and divergence directly by a big-step semantics, without resorting to a transition relation: we rely on the above described transition relation on incomplete derivations only to prove that the constructions are correct.

Corules (cf. Chapter 3) are crucial when defining extended big-step se-

mantics, explicitly modelling infinite computations. Indeed, in this case, standard induction is obviously not enough, while coinduction allows the derivation of spurious judgements, hence corules provides us with the possibility of refining the coinductive interpretation to get the expected semantics. Indeed, these constructions generalise concrete examples of big-step semantics modelling infinite computations by corules provided by Ancona, Dagnino, and Zucca (2017c, 2018).

More in detail, Chapter 5 studies standard big-step semantics, while Chapter 6 takes into account the extension with observations. In Chapter 5, we provide constructions explicitly modeling divergence (by traces and a special result ∞) and stuck computation (by a special result wrong, as described by Pierce (2002)). Then, relying on such constructions, we show how to state soundness of a predicate, that is, configurations satisfying the predicate cannot go wrong, and we describe a proof technique to prove soundness, based on three sufficient conditions on standard big-step rules, proving its correctness. In Chapter 6, in addition to problems related to big-step semantics, we have to face another issue: we need a way to abstractly model possibly infinite behaviour, namely, observations produced by infinite computations. To this end, we consider an algebraic structure, *ω-monoids*, which provides all the needed ingredients: notably, they have an infinite product, used to combine together the infinitely many observations of an infinite sequence of labelled transition steps, and a mixed product, used to combine finite and possibly infinite observations in big-step rules.

7.1 Related work

The research presented in this part follows a stream of work dating back to Cousot and Cousot (1992), who proposed a stratified approach, investigated by Leroy and Grall (2009) as well, with a separate judgment for divergence, defined coinductively. In this way, however, there is no unique formal definition of the behaviour of the modelled system and, furthermore, this cannot be smoothly extended to semantics with observations as, depending on their structure, even in a stratified definition there could be spurious judgements. An alternative possibility, also investigated by Leroy and Grall (2009), is to interpret coinductively the standard big-step rules (*coevaluation*). Unfortunately, coevaluation is non-deterministic, allowing the derivation of spurious judgements, and, thus, may fail to correctly capture the infinite behavior of a configuration: a diverging term, such as Ω, evaluates to any value, hence it cannot be properly distinguished from converging terms. Furthermore, in coevaluation there are still configurations, such as $\Omega\,(0\;0)$, for which no judgment can be derived, here because no judgment can be derived for the subterm $0\;0$; basically, this is due to the fact that divergence of a premise should be propagated and this cannot be correctly handled by coevaluation as divergence is not explicitly modelled.

Pretty big-step semantics by Charguéraud (2013) handles the issue of duplic-

ation of meta-rules by a unified judgment with a unique set of (meta-)rules and divergence modelled by a special value. Rules are interpreted coinductively, hence they allow the derivation of spurious judgements, but, thanks to the use of a special value for divergence and the particular structure of rules, they can solve most of the issues of coevaluation. However, this particular structure of rules is not as natural as usual big-step rules and, more importantly, it requires the introduction of new specific syntactic forms representing intermediate computation steps, as in small-step semantics, hence making the big-step semantics less abstract. This may be a problem, for instance, when proving soundness of a type system, as discussed in Chapter 5, as such intermediate configurations may be ill-typed.

Poulsen and Mosses (2017) subsequently present *flag-based big-step semantics*, which further streamlines the approach by combining it with the M-SOS technique (modular structural operational semantics), thereby reducing the number of (meta-)rules and premises, avoiding the need for intermediate configurations. The key idea is to extend configurations and results by flags explicitly modelling convergence and divergence, used to properly handle divergence propagation. To model divergence, they interpret rules coinductively, hence they allow the derivation of spurious judgements.

Differently from all the previously cited papers, which consider specific examples, the work by Ager (2004) shares with us the aim of providing a generic construction to model non-termination, basing on an arbitrary big-step semantics. Ager considers a class of big-step semantics identified by a specific shape of rules, and defines, in a small-step style, a proof-search algorithm which follows the big-step rules; in this way, converging, diverging and stuck computations are distinguished. This approach is somehow similar to our transition relation on partial evaluation trees, even tough the transition system we propose is directly defined on evaluation trees.

Ancona, Dagnino, and Zucca (2017c) firstly show that with corules one can define a unified big-step judgment with a unique set of rules avoiding spurious evaluations. This can be seen as *constrained coevaluation*. Indeed, corules add constraints on the infinite derivations to filter out spurious results, so that, for diverging terms, it is only possible to get ∞ as result. This is extended to include observations as traces by Ancona, Dagnino, and Zucca (2018). In this case, the effect of spurious evaluations can be more detrimental; indeed, when a diverging computation produces a finite observation o, coevaluation returns any observation of shape $o *^{\mathsf{m}} o_\infty$, and, thus, fails to correctly specify the effects of a non-terminating computation, as discussed in Chapter 6. In comparison to these works, here we provide a recipe for a *fully systematic* approach, and, furthermore, we allow reasoning on a *more general notion of observation*: at our knowledge, there is no other operational semantics framework able to capture divergence in conjunction with a notion of observation not limited to traces, as shown in Section 6.4.

Other proposals, by Owens et al. (2016) and Amin and Rompf (2017), are inspired by *definitional interpreters* (Reynolds, 1972), based on a step-indexed

approach (a.k.a."fuel"-based semantics) where computations are approximated to some finite amount of steps (typically with a counter); in this way divergence can be modeled by induction. Owens et al. (2016) investigates functional big-step semantics for proving by induction compiler correctness. Amin and Rompf (2017) explore inductive proof strategies for type soundness properties for the polymorphic type systems $F_{<:}$, and equivalence with small-step semantics. An inductive proof of type soundness for the big-step semantics of a Java-like language is proposed by Ancona (2014).

Conditional coinduction is employed by Danielsson (2010) to combine induction and coinduction in definitions of total parser combinators. Danielsson (2012), inspired by Leroy and Grall (2009), relying on the coinductive partiality monad, defines big-step semantics for λ-calculi and virtual machines as total, computable functions able to capture divergence.

Coinductive trace semantics in big-step style have been studied by Nakata and Uustalu (2009, 2010a,b). Their investigation started with the semantics of an imperative While language with no I/O (Nakata and Uustalu, 2009) where traces are possibly infinite sequences of states; semantic rules are all coinductive and define two mutually dependent judgments. Based on such a semantics, they define a Hoare logic (Nakata and Uustalu, 2010a); differently to our approach, weak bisimilarity between traces is required for proving that programs exhibit equivalent observable behaviors. This is due to the fact that "silent effects" (that is, non-observable internal steps) must be explicitly represented to guarantee guardedness conditions which ensure productivity of co-recursive definitions. This problem is overcome with corules in generalized inference systems.

This semantics has been subsequently extended with interactive I/O (Nakata and Uustalu, 2010b), by exploiting the notion of resumption monad: a tree representing possible runs of a program to model its non-deterministic behavior due to input values. Also in this case a big-step trace semantics is defined with two mutually recursive coinductive judgments, and weak bisimilarity is needed; however, the definition of the observational equivalence is more involved, since it requires nesting inductive definitions in coinductive ones. A generalised notion of resumption has been introduced later by Piróg and Gibbons (2014) in a category-theoretic and coalgebraic context.

Nakata and Uustalu (2009, 2010a) equivalence of the big-step and small-step semantics is proved; expressions and statements are distinct, and expressions cannot diverge. This is another significant difference with the languages considered in this part; for instance, the semantics of `out` e becomes simpler if e is forced to terminate, since the corresponding corule could be turned into a coaxiom, removing the premise, as it always holds.

The resumption monad of Nakata and Uustalu (2010b) and the partiality monad of Danielsson (2012) are inspired by the seminal work of Capretta (2005) on the *delay monad*, where coinductive types are exploited to model infinite computations by means of a type constructor for partial elements, which allows the formal definition of convergence and divergence and a type-

theoretic representation of general recursive functions; this type constructor is proved to constitute a strong monad, upon which subsequent related papers (Abel and Chapman, 2014; McBride, 2015; Chapman, Uustalu, and Veltri, 2019) elaborated to define other monads for managing divergence. In particular, McBride (2015) has proposed a more general approach based on a free monad for which the delay monad is an instantiation obtained through a monad morphism. All these proposals are based on the step-indexed approach.

More recently, interaction trees (ITrees) (Xia et al., 2020) have been presented as a coinductive variant of free monads with the main aim of defining the denotational semantics for effectful and possibly nonterminating computations, to allow compositional reasoning for mutually recursive components of an interactive system with fully mechanized proofs in Coq. Interaction trees are coinductively defined trees which directly support a more general fixpoint combinator which does need a step-indexed approach, as happens for the general monad of McBride. A `Tau` constructor is introduced to represent a silent step of computation, to express silently diverging computations without violating Coq's guardedness condition; as a consequence, generic definition of weak bisimulation on ITrees is required to remove any finite number of Tau's, similarly as happens in the approach of Nakata and Uustalu.

Finally, the notion of ω-monoid is a variation of the more standard ω-semigroups used in algebraic language theory (Perrin and Pin, 2004). Algebraic structures with a similar aim are proposed more often in the context of type systems, where the notion corresponding to observation is called *effect*. Such effect systems are traditionally commutative, hence effects typically form a bounded join semilattice, where the join is used to overapproximate different execution branches. This allows general formulations to study common properties, see, *e.g.*, the work by Marino and Millstein (2009). More recently, *sequential* effect systems have been proposed by Tate (2013), which take into account evaluation order as in our case. However, differently from our observations, effects in type systems do not need to be "infinite", since they model static approximations.

7.2 Future work

There are several directions for further research. First of all, in the context of big-step semantics with observations, we plan to explore refinements of the proposed construction to go beyond the determinism assumption needed in Section 6.5 to carry out the proof. Indeed, in presence of non-determinism, the current corule patterns fail to provide the correct semantics. In the introductory example of Section 6.1, extended with the non-deterministic choice $\oplus$, consider the expression $\Omega_c = \omega_c\,(\omega_c \oplus (\mathtt{out}\,0)$, with $\omega_c = \lambda x.x\,(x \oplus \mathtt{out}\,0)$; clearly we

have the following infinite derivation for any possibly infinite sequence o_∞:

$$\dfrac{\omega_c \Rightarrow \langle \omega_c, \varepsilon \rangle \qquad \dfrac{\omega_c \Rightarrow \langle \omega_c, \varepsilon \rangle}{\omega_c \oplus (\mathtt{out}\ 0) \Rightarrow \langle \omega_c, \varepsilon \rangle} \qquad \dfrac{\vdots}{\Omega_c \Rightarrow \langle \infty, o_\infty \rangle}}{\Omega_c \Rightarrow \langle \infty, o_\infty \rangle}$$

We expect to be able to derive the judgement only for $o_\infty = \varepsilon$, but using corules we can also prove, for instance, $\Omega_c \Rightarrow \langle \infty, 0 \rangle$, as follows:

$$\dfrac{\omega_c \Rightarrow \langle \omega_c, \varepsilon \rangle \qquad \dfrac{\dfrac{0 \Rightarrow \langle 0, \varepsilon \rangle}{\mathtt{out}\ 0 \Rightarrow \langle 0, 0 \rangle}}{\omega_c \oplus (\mathtt{out}\ 0) \Rightarrow \langle 0, 0 \rangle} \qquad \dfrac{}{0\,(0 \oplus (\mathtt{out}\ 0)) \Rightarrow \langle \infty, \varepsilon \rangle}}{\Omega_c \Rightarrow \langle \infty, 0 \rangle}$$

To avoid this, a possibility is to explicitly add in the judgment the partially obtained observation, that is, the product of the observations of converging premises. In this way, in the above example we could not derive 0 as observation, because in the infinite derivation all converging premises produce an empty sequence of values. Another possibility is to consider more sophisticated semantics of corules, as mentioned in Section 4.2, to avoid the fact that in finite derivations with corules we can use judgements, such as $0\,(0 \oplus (\mathtt{out}\ 0)) \Rightarrow \langle \infty, \varepsilon \rangle$, which are not derivable even in the coinductive interpretation of rules.

Another interesting direction is to study other approaches to model divergence in big-step semantics using our general meta-theory, that is, defining yet other constructions, such as adding a counter and timeout, as done by Owens et al. (2016) and Amin and Rompf (2017), or adding flags, as done by Poulsen and Mosses (2017). This would provide a general account of these approaches, allowing to study their properties in general, abstracting away particular features of concrete languages. A further direction is to consider other computational models such as probabilistic computations, which are quite difficult to model in big-step style, as shown by Dal Lago and Zorzi (2012). A possible starting point would be to adapt results for semantics with observations, viewing probabilities as a special kind of observations.

Concerning proof techniques for soundness, we also plan to compare our proof technique with the standard one for small-step semantics: if a predicate satisfies progress and subject reduction with respect to a small-step semantics, does it satisfy our soundness conditions with respect to an equivalent big-step semantics? To formally prove such a statement, the first step will be to express equivalence between small-step and big-step semantics, and such equivalence has to be expressed at the level of big-step rules, as it needs to be extendible to stuck and infinite computations. Note that, as a by-product, this will provide us with a proof technique to show equivalence between small-step and big-step semantics. Ancona et al. (2020a) make a first attempt to express such an equivalence for a more restrictive class of big-step semantics. On the other hand, the converse does not hold, as shown by the examples in Section 5.6.2 and Section 5.6.4.

Furthermore, it would be interesting to extend such techniques for soundness to big-step semantics with observation, taking inspiration from type and effect systems (Marino and Millstein, 2009; Tate, 2013).

Last but not least, to support reasoning by our framework on concrete examples, such as those in Sections 5.6 and 6.4, it is desirable to have a mechanisation of our meta-theory and related techniques. A necessary preliminary step in this direction is to provide support for corules in proof assistants, such as Agda or Coq, as discussed in Section 4.2.

PART III

Flexible regular coinduction

8

Regular coinduction by inference systems

As we have seen, inference systems are a powerful and fairly simple framework to structure and reason about possibly recursive definitions of predicates. They support both inductive and coinductive reasoning in a pretty natural way: in inductive reasoning we are only allowed to use finite derivations, while in the coinductive one we can prove judgements by arbitrary, finite or infinite, derivations.

Allowing infinite derivations makes coinductive reasoning very powerful: it makes it possible to derive judgements which require the proof of infinitely many different judgements. For instance, consider the following inference system used to prove that a stream (infinite sequence) contains only positive elements:

$$\frac{\mathsf{allPos}(s)}{\mathsf{allPos}(x{:}s)}\ x > 0$$

To prove that the stream of all odd natural numbers contains only positive elements, we can use the following infinite derivation:

$$\cfrac{\cfrac{\cfrac{\vdots}{\mathsf{allPos}(5{:}7{:}9{:}\ldots)}}{\mathsf{allPos}(3{:}5{:}7{:}\ldots)}}{\mathsf{allPos}(1{:}3{:}5\ldots)}$$

which is correct in coinductive reasoning and contains infinitely many different judgements.

However, there are cases where, even though we need an infinite derivation, this derivation requires only the proof of *finitely many* different judgements. This is often the case when dealing with cyclic structures, such as graphs or cyclic streams, since they are non-well-founded, but finitely representable. For instance, if we want to prove that the stream of all 1's contains only positive elements, we can use the following derivation:

$$\cfrac{\cfrac{\cfrac{\vdots}{\mathsf{allPos}(1{:}1{:}1{:}\ldots)}}{\mathsf{allPos}(1{:}1{:}1{:}\ldots)}}{\mathsf{allPos}(1{:}1{:}1{:}\ldots)}$$

which is infinite, but requires only the proof of $\mathsf{allPos}(1{:}1{:}1{:}\ldots)$.

Borrowing the terminology from trees (Courcelle, 1983), we call a derivation requiring the proof of finitely many different judgments *regular* (a.k.a. *rational*[1]), and we call *regular coinduction* (or *regular reasoning*) the approach that allows only regular derivations.

Whereas inductive and coinductive reasoning have well-known semantic foundations and proof principles, at our knowledge regular reasoning by means of inference rules has not been explored at the same extent. The aim of this chapter is to fill this gap, by providing solid foundations also to the regular approach. Indeed, we believe that the regular approach provides a very interesting middle way between induction and coinduction.

Indeed, inductive reasoning is restricted to finite derivations, but, in return, we implicitly get an (abstract) algorithm[2] which looks for a derivation of a judgement. Such an algorithm is sound and complete with respect to derivable judgements. That is, it may not terminate for judgements that do not have a finite derivation, but it is guaranteed to successfully terminate, finding a finite derivation, for all and only derivable judgments. Instead, coinductive reasoning allows also infinite derivations, but there is no hope, in general, to find an algorithm which succesfully terminates for derivable judgments, because, as we have seen, a derivation may require infinitely many different judgements to be proved[3].

Regular reasoning combines advantages of the two approaches: on one hand, it is not restricted to finite derivations, going beyond limits of induction, but, on the other hand, it still has, like induction, a finite nature, hence it is possible to design an algorithm which finds a derivation for all and only derivable judgments, as we will show in the following.

In detail, in this chapter we prove the following result about regular reasoning by inference systems. First, we provide an equivalent *model-theoretic* characterization of judgements derivable by a regular proof tree, showing it is an instance of the *rational fixed point*, defined by Adámek, Milius, and Velebil (2006). This is important since it provides a purely semantic view of regular coinduction. Moreover, from this we get a proof principle, the *regular coinduction principle*, which can be used to prove completeness of a set of inference rules against a set of correct judgements, that is, that all correct judgements are derivable by a regular proof tree.

Then, we provide another equivalent *inductive* characterization of judgements derivable by a regular proof tree. Following the structure of the operational model of coinductive logic programming (Simon et al., 2006; Ancona and Dovier, 2015), but in the purely semantic setting of inference systems, we enrich judgements by a finite set of *circular hypotheses*, used to keep track of

1 The terms regular and rational are synonyms. However we will mainly use the second one for the model-theoretic approach, see Section 8.2.

2 In this chapter we use the word "algorithm" to indicate a procedure which is not required to terminate.

3 This is just an intuitive explanation. This fact has been proved by Ancona and Dovier (2015) for logic programs, which are a particular, syntactic, instance of general inference systems considered in this chapter.

already encountered judgements, so that, when the same judgement is found again, it can be used as an axiom. This nicely formalizes, by an abstract construction in the general setting of inference systems and a correctness proof given once and for all, techniques used in different specific cases for dealing with cyclic structures inductively, by detecting cycles to ensure termination. Furthermore, this provides us with a sound and complete algorithm to find a regular derivation for a judgment, if any. Finally, relying on the inductive characterization, we define a proof technique to show soundness of a set of inference rules against a set of correct judgements, that is, that all derivable judgements are correct.

Moreover, we show that all these results can be smoothly extended to flexible coinduction by inference systems with corules, as described in Chapter 3, thus enabling *flexible regular coinduction*.

The rest of the chapter is organised as follows. In Section 8.1 we introduce the regular interpretation in proof-theoretic terms. In Section 8.2 we define the rational fixed point in a lattice-theoretic setting, and in Section 8.3 we prove that the regular interpretation coincides with a rational fixed point. Section 8.4 provides the equivalent inductive characterization of the regular interpretation and Section 8.5 discusses proof techniques for regular reasoning. In Section 8.6 we extend all the previously presented results to flexible coinduction.

8.1 Inference systems and regular derivations

In this section we introduce the regular interpretation of inference systems illustrating it by some examples. For basic notions about inference systems we refer to Chapter 2.

Le us assume a universe $\mathcal{U}$ and an inference system on $\mathcal{U}$. Throughout this chapter we will assume inference systems to be *finitary* (cf. Definition 2.30), namely, all rules have a finite set of premises. Under this assumption, well-founded proof trees are always finite and infinite proof trees are always non-well-founded, hence we will use this simpler terminology.

In the coinductive interpretation (cf. Definition 2.6), we allow arbitrary proof trees, hence we can derive judgements requiring infinitely many different judgements to be proved. However, there are cases where we still need infinite derivations, but only of finitely many judgements. This idea of an infinite proof tree containing only finitely many different judgements nicely corresponds to a well-known class of trees: *regular trees* (Courcelle, 1983). We say that a tree is *regular* if it has a finite number of different subtrees. Then, we can define another set of judgements:

DEFINITION 8.1 (Regular interpretation): The *regular interpretation* of an inference system $\mathcal{I}$ is the set $\rho[\![\mathcal{I}]\!]$ of judgements having a regular proof tree.

In the following we will write $\mathcal{I} \vdash_\rho j$ for $j \in \rho[\![\mathcal{I}]\!]$. To ensure that the regular interpretation is well-defined, that is, it is an $\mathcal{I}$-interpretation (cf. Defini-

tion 2.2), we have to check it is both $\mathcal{I}$-closed and $\mathcal{I}$-consistent, or, equivalently, a fixed point of $F_{\mathcal{I}}$. We refer to Section 8.3 for this proof.

Let us illustrate regular proof trees by an example. Recall the last example in Section 2.1.2 (cf. p.18) , defining the judgement $\mathsf{dist}_G(v, u, \delta)$, where G is a graph, v and u are nodes in G and $\delta \in \mathbb{N} \cup \{\infty\}$, stating that the distance from v to u in G is δ. As already done, we represent a graph by the adjacency function $G : V \to \wp(V)$, where V is the finite set of nodes. We report below the (meta-)rules, where we assume $\min \emptyset = \infty$:

$$(\textsc{empty})\ \frac{}{\mathsf{dist}_G(v, v, 0)}$$

$$(\textsc{adj})\ \frac{\mathsf{dist}_G(v_1, u, \delta_1) \quad \dots \quad \mathsf{dist}_G(v_n, u, \delta_n)}{\mathsf{dist}_G(v, u, 1 + \min\{\delta_1, \dots, \delta_n\})} \quad \begin{array}{l} v \neq u \\ G(v) = \{v_1, \dots, v_n\} \end{array}$$

As already discussed in Section 2.1.2, the inductive interpretation is not enough as it cannot deal with cycles, hence, we need infinite derivations, but, since the set of nodes is finite, to compute the distance we need only finitely many judgements (one judgement for each node), thus regular derivations should be enough. Indeed, the infinite derivation shown in Figure 2.2, is actually regular.

As we said, standard inductive and coinductive interpretations are fixed points of the inference operator. In the next few sections, we will show that this is the case also for the regular interpretation.

8.2 The rational fixed point

In this section we define the rational fixed point in a lattice-theoretic setting, which will be the basis for the fixed point characterisation of the regular interpretation. The construction we present in Definition 8.3 and Theorem 8.4 is an instance of analogous constructions developed by Adámek, Milius, and Velebil (2006) and Milius, Pattinson, and Wißmann (2016, 2019) in a more general category-theoretic setting. We rephrase these notions and results in the lattice-theoretic setting, since this is enough for the aim of this chapter and definitions and proofs are simpler and understandable by a wider audience.

For basic definitions on complete lattices we refer to Section 2.2. Assume a complete lattice $\langle L, \sqsubseteq \rangle$. An element $x \in L$ is *compact* if, for all $A \subseteq L$ such that $x \sqsubseteq \bigsqcup A$, there is a finite subset $B \subseteq A$ such that $x \sqsubseteq \bigsqcup B$. We denote by $C(L)$ the set of compact elements in L. It is easy to check that $C(L)$ is closed under binary joins, that is, if $x, y \in L$ are compact, then $x \sqcup y$ is compact as well. The paradigmatic example of complete lattice is the power-set $\wp(X)$ of a set X, namely, the set of all subsets of X ordered by set inclusion. In the power-set lattice $\langle \wp(X), \subseteq \rangle$, compact elements are finite subsets of X.

An *algebraic lattice* is a complete lattice $\langle L, \sqsubseteq \rangle$ where each $x \in L$ is the join of all compact elements below it, that is, $x = \bigsqcup\{y \in C(L) \mid y \sqsubseteq x\}$. In other words, an algebraic lattice is generated by the set of its compact

elements, since each element can be decomposed as a (possibly infinite) join of compact elements. The power-set lattice is algebraic, since each element can be decomposed as a union of singletons, which are obviously compact.

In previous chapters we have considered monotone functions over complete lattices, as monotonicity was enough to construct least and greatest fixed points. Here, we are interested in a different class of functions, called finitary functions, defined below. A subset $A \subseteq L$ is *directed* if, for all $x, y \in A$, there is $z \in A$ such that $x \sqsubseteq z$ and $y \sqsubseteq z$.

DEFINITION 8.2 : A function $F : L \to L$ is *finitary* if it preserves least upper bounds of all directed subsets of L, that is, for each directed subset $A \subseteq L$, $F(\bigsqcup A) = \bigsqcup F_!(A)$.

A finitary function is also monotone: if $x \sqsubseteq y$ then the set $\{x, y\}$ is directed and its join is y, hence we get $F(y) = F(x) \sqcup F(y)$, that is, $F(x) \sqsubseteq F(y)$. As a consequence, the Knaster-Tarski theorem (cf. Theorem 2.10) applies also to finitary functions, which hence admit least and greatest fixed points. We will show that for a finitary function over an algebraic lattice we can construct another fixed point lying between the least and the greatest one. In the following we assume an algebraic lattice $\langle L, \sqsubseteq \rangle$.

DEFINITION 8.3 : Let $F : L \to L$ be a finitary function. The *rational fixed point* of F, denoted by ρF, is the join of all compact post-fixed points of F, that is, if $R_F = \{x \in C(L) \mid x \sqsubseteq F(x)\}$,

$$\rho F = \bigsqcup R_F$$

Note that, since both compact elements and post-fixed points are closed under binary joins, we have that, for all $x, y \in R_F$, $x \sqcup y \in R_F$, but, in general, ρF is not compact, because it is the join of an infinite set.

The following theorem ensures that the rational fixed point is well-defined, that is, it is indeed a fixed point. Such result is a consequence of a general category-theoretic analysis (Adámek, Milius, and Velebil, 2006), we rephrase the proof in our more specific setting as it is much simpler.

THEOREM 8.4 : Let $F : L \to L$ be a finitary function, then ρF is a fixed point of F.

Proof: Since ρF is defined as the least upper bound of a set of post-fixed points, it is post-fixed as well. Hence, we have only to check that $F(\rho F) \sqsubseteq \rho F$.

First, since L is algebraic we have $F(\rho F) = \bigsqcup\{x \in C(L) \mid x \sqsubseteq F(\rho F)\}$, hence it is enough to prove that, for all $x \in C(L)$ such that $x \sqsubseteq F(\rho F)$, we have $x \sqsubseteq \rho F$. Consider $x \in C(L)$ such that $x \sqsubseteq F(\rho F)$. Note that R_F is a directed set, indeed, if $X \subseteq R_F$ is finite, then $\bigsqcup X \in R_F$, hence, since F is finitary, we have $F(\rho F) = F(\bigsqcup R_F) = \bigsqcup F_!(R_F)$. Therefore, $x \sqsubseteq \bigsqcup F_!(R_F)$ and, since x is compact, there is a finite subset $W \subseteq R_F$ such that $x \sqsubseteq \bigsqcup F_!(W)$. Set $w = \bigsqcup W$, since F is monotone, we get $x \sqsubseteq \bigsqcup F_!(W) \sqsubseteq F(\bigsqcup W) = F(w)$. By definition, $w \in R_F$, namely, it is compact and post-fixed, hence we get

$x \sqcup w \sqsubseteq F(w) \sqsubseteq F(x \sqcup w)$, since F is monotone. Finally, $x \sqcup w$ is compact, as it is the join of compact elements, hence $x \sqcup w \in R_F$, and this implies $x \sqsubseteq x \sqcup w \sqsubseteq \rho F$, as needed. □

As for least and greatest fixed points, as an immediate consequence of Definition 8.3, we get a proof principle to show that an element is below ρF.

PROPOSITION 8.5 : Let $F : L \to L$ be a finitary function and $z \in L$, then, if there is a set $X \subseteq C(L)$ such that

- for all $x \in X$, $x \sqsubseteq F(x)$, and
- $z \sqsubseteq \bigsqcup X$,

then $z \sqsubseteq \rho F$.

Proof: If these conditions hold, then we have $X \subseteq R_F$, hence $z \sqsubseteq \bigsqcup X \sqsubseteq \bigsqcup R_F = \rho F$. □

8.3 Fixed point semantics for regular coinduction

In this section, we prove that the regular interpretation $\rho[\![\mathcal{I}]\!]$ of a (finitary) inference system $\mathcal{I}$ (cf. Definition 8.1) coincides with the rational fixed point of the inference operator $F_{\mathcal{I}}$. Rather than giving an ad-hoc proof, we exploit the general framework presented in Section 2.3, extending results for least and greatest fixed point to the rational one. Assume a finitary inference system $\mathcal{I}$ on the universe $\mathcal{U}$.

First of all, we have to express more formally the proof-theoretic semantics. We refer to Sections 2.1 and 2.1.1 for formal definitions of trees and proof trees. Recall from these sections that, given a tree τ on the set A, $\mathsf{N}(\tau) \subseteq A^\star$ is the set of nodes of τ and $\mathsf{r}(\tau) \in A$ is the root of τ. Further, $\mathsf{SubTr}(\tau)$ is the set of all subtrees of τ and $\mathsf{dst}(\tau) \subseteq \mathsf{SubTr}(\tau)$ is the set of direct subtrees of τ. Therefore, a tree τ is *regular* iff $\mathsf{SubTr}(\tau)$ is finite.

From Section 2.3, recall also that $\mathsf{r} : \mathcal{T}_{\mathcal{U}} \to \mathcal{U}$ is the function mapping a tree to its root, and $\mathsf{r}_! : \wp(\mathcal{T}_A) \to \wp(A)$ and $\mathsf{r}^* : \wp(A) \to \wp(\mathcal{T}_A)$ are the direct image and the inverse image functions, respectively. There is an obvious adjunction $\mathsf{r}_! \dashv \mathsf{r}^*$.

We have already proved that, when $\mathcal{I}$ is finitary, the inference operator $F_{\mathcal{I}}$ is not only monotone, but upward ω-continuous (cf. Theorem 2.31). However, both the inference operator $F_{\mathcal{I}}$ and the tree inference operator $T_{\mathcal{I}}$, when $\mathcal{I}$ is finitary, have a much stronger property: they are finitary (cf. Definition 8.2), that is, preserve least upper bounds of all directed sets.

PROPOSITION 8.6 : If $\mathcal{I}$ is finitary, then $F_{\mathcal{I}}$ and $T_{\mathcal{I}}$ are finitary.

Proof: We do the proof for $T_{\mathcal{I}}$. Let $X \subseteq \wp(\mathcal{U})$ be a directed subset. Since $T_{\mathcal{I}}$ is monotone, we have $\bigcup T_{\mathcal{I}!}(X) \subseteq T_{\mathcal{I}}(\bigcup X)$, hence we have only to check the other inclusion. If $\tau \in T_{\mathcal{I}}(\bigcup X)$, $\mathsf{dst}(\tau) \subseteq \bigcup X$ and, since $\mathsf{dst}(\tau)$ is finite, as it is

in bijection with a set of premises, which must be finite as $\mathcal{I}$ is finitary, there is a finite subset $Y \subseteq X$ such that $\mathsf{dst}(\tau) \subseteq \bigcup Y$. Then, since X is directed, there is $A \in X$ such that $\bigcup Y \subseteq A$, thus $\mathsf{dst}(\tau) \subseteq A$, and so $\tau \in T_{\mathcal{I}}(A) \subseteq \bigcup T_{\mathcal{I}!}(X)$, as needed. □

Thanks to Proposition 8.6 and Theorem 8.4, and because power-set lattices are algebraic, we know that rational fixed points $\rho F_{\mathcal{I}}$ and $\rho T_{\mathcal{I}}$ are both well-defined. As happens for the least and greatest fixed points of $T_{\mathcal{I}}$, which coincide with the well-founded and arbitrary proof trees, respectively (Lemmas 2.20 and 2.21), we show that the rational fixed point of $T_{\mathcal{I}}$ coincides with the set of regular proof trees. To this end, we have the following characterization of post-fixed points of $T_{\mathcal{I}}$.

LEMMA 8.7 : Let $X \subseteq \mathcal{T}_{\mathcal{U}}$ be a set of trees on $\mathcal{U}$. If $X \subseteq T_{\mathcal{I}}(X)$, then

$$X = \bigcup_{\tau \in X} \mathsf{SubTr}(\tau)$$

Proof: We start from the left-to-right implication. The inclusion $\subseteq$ is trivial, since $\tau \in \mathsf{SubTr}(\tau)$ for any tree τ. To prove the other inclusion, set $\tau \in X$, we have to show that, for all $\alpha \in \mathsf{N}(\tau)$, $\tau_{|\alpha} \in X$. The proof is by induction on α. If α is empty, then $\tau_{|\alpha} = \tau \in X$ by hypothesis. If $\alpha = \beta j$, then $\tau_{|\beta j} = (\tau_{|\beta})_{|_j}$ and $\tau_{|\beta} \in X$ by induction hypothesis. Since $X \subseteq T_{\mathcal{I}}(X)$, we have $(\tau_{|\beta})_{|_j} \in \mathsf{dst}(\tau_{|\beta}) \subseteq X$, as needed. □

LEMMA 8.8 : $\rho T_{\mathcal{I}}$ is the set of regular proof trees in $\mathcal{I}$.

Proof: Let τ be a regular tree, then $\mathsf{SubTr}(\tau)$ is finite and, by Lemma 2.18 (2), it is a post-fixed point of $T_{\mathcal{I}}$. Hence, $\tau \in \mathsf{SubTr}(\tau) \subseteq \rho T_{\mathcal{I}}$, by Proposition 8.5, as needed.

Let now $X \subseteq \mathcal{T}_{\mathcal{U}}$ be a finite post-fixed point of $T_{\mathcal{I}}$, then we just have to show that all $\tau \in X$ are regular proof trees. Let $\tau \in X$, by Lemma 2.18 (1), τ is a proof tree and, by Lemma 8.7, we have $\mathsf{SubTr}(\tau) \subseteq X$, hence $\mathsf{SubTr}(\tau)$ is finite, that is, τ is regular, as needed. □

Thanks to Lemma 8.8 and Definition 8.1, we can express the definition of the regular interpretation by the equality $\rho[\![\mathcal{I}]\!] = \mathsf{r}_!(\rho T_{\mathcal{I}})$.

Then, similarly to results for least and greatest fixed points (cf. Theorems 2.23 and 2.24), the theorem we have to prove is the following:

THEOREM 8.9 : $\rho[\![\mathcal{I}]\!] = \mathsf{r}_!(\rho T_{\mathcal{I}}) = \rho F_{\mathcal{I}}$.

The proof follows the same strategy of the one for the greatest fixed point and the coinductive interpretation, which relied on Theorem 2.4, stating a general property of trees. Therefore, we need to show a similar property for regular trees. To this end, assume a set A and denote by $\mathcal{R}_A$ the set of regular trees on A. Note that functions $\mathsf{dst} : \mathcal{T}_A \to \wp(\mathcal{T}_A)$ and $\mathsf{r} : \mathcal{T}_A \to A$, mapping a

tree to its direct subtrees and to its root, respectively, restrict to $\mathcal{R}_A$, because subtrees of a regular tree are regular as well. Basically, we show that, starting from a graph structure on a finite subset of A, for each node of the graph there is a unique way to construct a tree coherent with the graph structure, which is also regular. In this context a graph is a function $g : X \to \wp(X)$, modelling the adjaciency function, that is, X is the set of nodes and, for all $x \in X$, $g(x)$ is the set of adjacients of x.

THEOREM 8.10 : Let $g : X \to \wp(X)$ be a function, with X finite, and $v : X \to A$ be an injective function. Then, there exists a unique function $p : X \to \mathcal{R}_A$ such that the following diagram commutes:

$$\begin{array}{ccc} X & \xrightarrow{\quad p \quad} & \mathcal{R}_A \\ {\scriptstyle \langle g, v \rangle}\downarrow & & \downarrow{\scriptstyle \langle \mathsf{dst}, \mathsf{r} \rangle} \\ \wp(X) \times A & \xrightarrow{p_! \times \mathsf{id}A} & \wp(\mathcal{R}_A) \times A \end{array}$$

Proof: By Theorem 2.4, we know that there is a unique function $p : X \to \mathcal{T}_A$ such that $\langle \mathsf{dst}, \mathsf{r} \rangle \cdot p = (p_! \times \mathsf{id}_A) \cdot \langle g, v \rangle$, and this function is injective. Hence, we have only to show that p corestricts to $\mathcal{R}_A$, that is, $p(x)$ is regular, for all $x \in X$. We prove, by induction on $\alpha \in \mathsf{N}(p(x))$, that, for all $\alpha \in \mathsf{N}(p(x))$, there exists $y \in X$ such that $p(x)_{|\alpha} = p(y)$.

Case: ε We have $p(x)_{|\varepsilon} = p(x)$, as needed.

Case: βa We have $p(x)_{|\beta a} = (p(x)_{|\beta})_{|a}$ and, by induction hypothesis, there is $z \in X$ such that $p(x)_{|\beta} = p(z)$. Therefore, we have $p(x)_{|\alpha} = p(z)_{|a} \in \mathsf{dst}(p(z))$, hence, since the diagram commutes, there is $y \in g(z) \subseteq X$ such that $p(z)_{|a} = p(y)$, as needed.

Therefore, for all $x \in X$, $\mathsf{SubTr}(p(x)) \subseteq p_!(X)$, which is finite as X is finite, hence $p(x)$ is regular. □

We can now prove Theorem 8.9.

Proof (Theorem 8.9)*:* By Lemma 8.8 we get $\rho[\![\mathcal{I}]\!] = \mathsf{r}_!(\rho T_{\mathcal{I}})$. Recall that in $\wp(\mathcal{T}_{\mathcal{U}})$, compact elements are finite subsets, hence the set of all compact elements is $\wp_\omega(\mathcal{T}_{\mathcal{U}})$. Then, by definition of the rational fixed point and since $\mathsf{r}_!$ preserves arbitrary unions (it is a left adjoint), we get $\mathsf{r}_!(\rho T_{\mathcal{I}}) = \bigcup\{\mathsf{r}_!(X) \mid X \in \wp_\omega(\mathcal{T}_{\mathcal{U}}) \text{ and } X \subseteq T_{\mathcal{I}}(X)\}$. Hence, if $X \in \wp_\omega(\mathcal{T}_{\mathcal{U}})$ and $X \subseteq T_{\mathcal{I}}(X)$, $\mathsf{r}_!(X)$ is obviously finite and, by Corollary 2.17, it is also a post-fixed point of $F_{\mathcal{I}}$. Therefore, by definition of the rational fixed point, we get $\mathsf{r}_!(X) \subseteq \rho F_{\mathcal{I}}$, and this proves $\mathsf{r}_!(\rho T_{\mathcal{I}}) \subseteq \rho F_{\mathcal{I}}$.

To conclude the proof, we show that $\rho F_{\mathcal{I}} \subseteq \mathsf{r}_!(\rho T_{\mathcal{I}})$. To this end, we just have to prove that, given a finite set $X \in \wp_\omega(\mathcal{U})$ such that $X \subseteq F_{\mathcal{I}}(X)$, each judgement $j \in X$ has a regular proof tree. Since $X \subseteq F_{\mathcal{I}}(X)$, X is consistent (Definition 2.2), that is, for each $j \in X$, there is $Pr_j \subseteq X$ such that $\langle Pr_j, j \rangle \in \mathcal{I}$. Hence, applying Theorem 8.10, where g maps j to Pr_j and

v is the restriction of the identity on $\mathcal{U}$ to X, we get an injective function $p : X \to \mathcal{R}_{\mathcal{U}}$, which makes the diagram in Theorem 8.10 commute. We have still to prove that $p(j)$ is a proof tree. To this end, by Lemma 2.18 (1), we just have to show that the set $p_!(X) = \{p(j) \mid j \in X\}$ is a post-fixed point of $T_{\mathcal{I}}$. By commutativity of the diagram, we have $\mathsf{dst}(p(j)) = p_!(g(j)) \subseteq p_!(X)$, $\mathsf{r}(p(j)) = j$ and $\mathsf{r}_!(\mathsf{dst}(p(j))) = Pr_j$, hence, as $\langle Pr_j, j\rangle \in \mathcal{I}$, we get $p(j) \in T_{\mathcal{I}}(p_!(X))$, as needed. □

8.4 An inductive characterization

Although the regular interpretation is essentially coinductive, as it allows non-well-founded derivations, it has an intrinsic finite nature, because it requires proof trees to be regular, that is, to have only finitely many subtrees. Given this finiteness, a natural question is the following: is it possible to finitely find a derivation for a judgement belonging to the regular interpretation? In this section we show this is the case, by providing an inductive characterization of the regular interpretation.

The idea behind such an inductive characterisation is simple. Regular trees are basically cyclic structures. Usually, to deal with cyclic structures inductively, we need to use auxiliary structures to detect cycles, to ensure termination. For instance, in order to perform a visit of a graph, we detect cycles by marking already encountered nodes. The inductive characterization described below models such cycle detection mechanism in an abstract and canonical way, in the general setting of inference systems. The idea is the following: during the proof, we keep track of already encountered judgements and, if we find again the same judgement, we can use it as an axiom.

This approach is intuitively correct, since in a regular proof tree there are only finitely many subtrees, hence infinite paths *must* contain repeated judgements, and this mechanism is designed precisely to detect such repetitions.

We now formally define the construction and prove its correctness. Let $\mathcal{I}$ be a finitary inference system on the universe $\mathcal{U}$. We consider judgements of shape $H \triangleright j$ where $H \subseteq \mathcal{U}$ is a finite set of judgements, called *circular hypotheses*, and $j \in \mathcal{U}$ is a judgement. Then, we have the following definition.

DEFINITION 8.11 : The inference system $\mathcal{I}^{\circlearrowleft}$ consists of the following rules:

$$(\text{HP})\ \frac{}{H \triangleright j}\ \ j \in H \qquad (\text{UNFOLD})\ \frac{H \cup \{j\} \triangleright j_1 \quad \dots \quad H \cup \{j\} \triangleright j_n}{H \triangleright j}\ \ \langle \{j_1, \dots, j_n\}, j\rangle \in \mathcal{I}$$

Therefore, in the system $\mathcal{I}^{\circlearrowleft}$, we have the same rules as in $\mathcal{I}$, that, however, extend the set of circular hypotheses by adding the conclusion of the rule as an hypothesis in the premises. Furthermore, $\mathcal{I}^{\circlearrowleft}$ has also an additional axiom that allows application of circular hypotheses.

The correctness of the construction in Definition 8.11 is expressed by the fact that a judgement j has a regular proof tree in $\mathcal{I}$ if and only if it has a finite derivation in $\mathcal{I}^{\circlearrowleft}$ without circular hypotheses, as formally stated by the next theorem.

THEOREM 8.12 : $\mathcal{I}^{\circlearrowleft} \vdash_\mu \emptyset \triangleright j$ iff $\mathcal{I} \vdash_\rho j$.

We prove a more general version of the theorem. First of all, if $X \subseteq \mathcal{U}$, we denote by $\mathcal{I}_{\oplus X}$ the system obtained from $\mathcal{I}$ by adding an axiom for each element of X, hence we have $F_{\mathcal{I}_{\oplus X}}(Y) = F_{\mathcal{I}}(Y) \cup X$, for all $Y \subseteq \mathcal{U}$. Then, the left-to-right implication of Theorem 8.12 is an immediate consequence of the following lemma:

LEMMA 8.13 : If $\mathcal{I}^{\circlearrowleft} \vdash_\mu H \triangleright j$ then $\mathcal{I}_{\oplus H} \vdash_\rho j$.

Proof: The proof is by induction on rules in $\mathcal{I}^{\circlearrowleft}$. There are two types of rules, hence we distinguish two cases:

(HP) we have $j \in H$, hence $\langle \emptyset, j \rangle \in \mathcal{I}_{\oplus H}$, thus $\mathcal{I}_{\oplus H} \vdash_\rho j$.

(UNFOLD) we have a rule $\langle \{j_1, \ldots, j_n\}, j \rangle \in \mathcal{I}$ and, by induction hypothesis, we get $\mathcal{I}_{\oplus H \cup \{j\}} \vdash_\rho j_i$, for all $i \in 1..n$. Since $\rho[\![\mathcal{I}_{\oplus H \cup \{j\}}]\!]$ is a rational fixed point (Theorem 8.9), for all $i \in 1..n$, there is a finite set X_i such that $j_i \in X_i \subseteq F_{\mathcal{I}_{\oplus H \cup \{j\}}}(X_i) = F_{\mathcal{I}_{\oplus H}}(X_i) \cup \{j\}$. Set $X = \bigcup_{i=1}^{n} X_i$, then X is finite, $X \subseteq F_{\mathcal{I}_{\oplus H}}(X) \cup \{j\}$, as $F_{\mathcal{I}_{\oplus H}}$ is monotone, and $j \in F_{\mathcal{I}}(X)$, because, by construction, $\{j_1, \ldots, j_n\} \subseteq X$. Thus, we get $X \cup \{j\} \subseteq F_{\mathcal{I}_{\oplus H}}(X) \cup \{j\} = F_{\mathcal{I}_{\oplus H}}(X)$, because $j \in F_{\mathcal{I}}(X) \subseteq F_{\mathcal{I}_{\oplus H}}(X)$, hence $X \cup \{j\}$ is a post-fixed point of $F_{\mathcal{I}_{\oplus H}}$, since it is monotone. Therefore, since $X \cup \{j\}$ is post-fixed and finite, by Proposition 8.5 and Theorem 8.9, we get $\mathcal{I}_{\oplus H} \vdash_\rho j$. □

The proof of the other implication relies on a family of auxiliary functions indexed over finite subsets of $\mathcal{U}$. For each $H \in \wp_\omega(\mathcal{U})$, the function tr_H takes a finite graph $g : X \to \wp(X)$, with $X \in \wp_\omega(\mathcal{U})$, a judgement $j \in X$ and a subset $S \subseteq X$ and returns a tree whose nodes are judgements of shape $H' \triangleright j'$. This function is recursively defined as follows:

$$\mathsf{tr}_H(g, j, S) = \begin{cases} \dfrac{}{H \cup S \triangleright j} & j \in H \cup S \\ \dfrac{\mathsf{tr}_H(g, j_1, S \cup \{j\}) \quad \ldots \quad \mathsf{tr}_H(g, j_n, S \cup \{j\})}{H \cup S \triangleright j} & j \notin H \cup S,\ g(j) = \{j_1, \ldots, j_n\} \end{cases}$$

The function tr_H enjoys the following properties:

PROPOSITION 8.14 : For all $H \in \wp_\omega(\mathcal{U})$, $g : X \to \wp(X)$ with $X \in \wp_\omega(\mathcal{U})$, $j \in X$ and $S \subseteq X$, $\mathsf{tr}_H(g, j, S)$ is defined.

Proof: Denote by $c(S)$ the cardinality of the set $X \setminus (H \cup S)$. We prove that, for all $n \in \mathbb{N}$ and $S \subseteq X$, if $c(S) = n$ then $\mathsf{tr}_H(g, j, S)$ is defined. The proof is by induction on n.

BASE If $c(S) = n = 0$, then $X \subseteq H \cup S$, hence $j \in H \cup S$ hence $\mathsf{tr}_H(g, j, S) = \dfrac{}{H \cup S \triangleright j}$.

INDUCTION If $c(S) = n + 1$, if $j \in H \cup S$ then $\mathsf{tr}_H(g, j, S)$ is defined as before; otherwise, we have $j \notin H \cup S$ and, if $g(j) = \{j_1, \ldots, j_k\}$,

then, for all $i \in 1..n$, $\mathrm{tr}_H(g, j_i, S \cup \{j\}) = \tau_i$ by induction hypothesis, as $c(S \cup \{j\}) = n$ since $j \notin S$, hence $\mathrm{tr}_H(g, j, S) = \dfrac{\tau_1 \quad \cdots \quad \tau_n}{H \cup S \triangleright j}$, as needed. □

PROPOSITION 8.15 : For all $H \in \wp_\omega(\mathcal{U})$, $g : X \to \wp(X)$ with $X \in \wp_\omega(\mathcal{U})$, $j \in X$ and $S \subseteq X$, if $\langle g(j'), j' \rangle \in \mathcal{I}$, for all $j' \in X \setminus H$, then $\mathrm{tr}_H(g, j, S)$ is a finite proof tree for $H \cup S \triangleright j$ in $\mathcal{I}^{\cup}$.

Proof: The proof is a straightforward induction on the definition of tr_H. □

We can now prove the following lemma, which concludes the proof of Theorem 8.12.

LEMMA 8.16 : If $\mathcal{I}_{\oplus H} \vdash_\rho j$ then $\mathcal{I}^{\cup} \vdash_\mu H \triangleright j$.

Proof: If $j \in \rho[\![\mathcal{I}_{\oplus H}]\!]$, since $\rho[\![\mathcal{I}_{\oplus H}]\!] = \rho F_{\mathcal{I}_{\oplus H}}$ (Theorem 8.9), we have that there exists a finite set $X \subseteq \mathcal{U}$ such that $j \in X \subseteq F_{\mathcal{I}_{\oplus H}}(X) = F_{\mathcal{I}}(X) \cup H$. Then, for each $j' \in X \setminus H$, there is $Pr_{j'} \subseteq X$ such that $\langle Pr_{j'}, j' \rangle \in \mathcal{I}$. Define $g : X \to \wp(X)$ by $g(j') = Pr_{j'}$, if $j' \in X \setminus H$, and $g(j') = \emptyset$ otherwise. Therefore, by Proposition 8.14, $\mathrm{tr}_H(g, j, \emptyset)$ is defined and, by Proposition 8.15, it is a finite proof tree for $H \triangleright j$ in $\mathcal{I}^{\cup}$, hence we get $\mathcal{I}^{\cup} \vdash_\mu H \triangleright j$, as needed. □

We conclude the section by discussing a more operational aspect of Definition 8.11. In this definition, we aimed at being as liberal as possible, hence the two types of rules are not mutually exclusive: for a judgement $H \triangleright j$ with $j \in H$ we can either apply the circular hypothesis or use a rule from $\mathcal{I}$. Since here we are only interested in derivability, this aspect is not so relevant, however, it becomes more interesting from an algorithmic perspective. Indeed, we can consider an alternative definition, where we allow the second type of rule only if $j \notin H$; in other words, we apply circular hypotheses as soon as we can.

In this way we would have less valid proof trees in $\mathcal{I}^{\cup}$, but the set of derivable judgements remains the same. Indeed, Lemma 8.13 ensures soundness also of the deterministic version, because any proof tree in the deterministic version is also a proof tree in the non-deterministic one. On the other hand, Lemma 8.16 ensures completeness of the deterministic version; indeed, the functions tr_H build a proof tree in the deterministic version, since they perform the additional check $j \notin H$ to apply rules from $\mathcal{I}$.

8.5 Regular reasoning

In this section we discuss proof techniques for regular reasoning, which can be defined thanks to the results proved in Section 8.3 and Section 8.4.

Let $\mathcal{I}$ be a finitary inference system on the universe $\mathcal{U}$. As discussed in Sections 2.4 and 3.4, we are typically interested in comparing the regular interpretation of $\mathcal{I}$ to a set of judgements $\mathcal{S} \subseteq \mathcal{U}$ (*specification*), focusing on

soundness ($\rho[\![\mathcal{I}]\!] \subseteq \mathcal{S}$) and *completeness* ($\mathcal{S} \subseteq \rho[\![\mathcal{I}]\!]$) properties. Proving both properties amounts to say that the inference system actually defines the given specification $\mathcal{S}$.

For completeness proofs, we rely on the fixed point characterization of $\rho[\![\mathcal{I}]\!]$ (Theorem 8.9). Indeed, since $\rho[\![\mathcal{I}]\!] = \rho F_{\mathcal{I}}$, we get a proof principle, rephrasing Proposition 8.5 as follows:

PROPOSITION 8.17 (Regular coinduction): Let $\mathcal{S} \subseteq \mathcal{U}$ be a set of judgements. If, for all $j \in \mathcal{S}$, there is a finite set $X \subseteq \mathcal{U}$ such that $j \in X \subseteq F_{\mathcal{I}}(X)$, then, $\mathcal{S} \subseteq \rho[\![\mathcal{I}]\!]$.

Proof: Immediate from Proposition 8.5. □

This looks very much like the usual coinduction principle, but it additionally requires X to be finite. The condition $X \subseteq F_{\mathcal{I}}(X)$ is equivalent to rquiring X to be $\mathcal{I}$-consistent (cf. Definition 2.2).

EXAMPLE 8.18 : To show how proofs by regular coinduction work, as first example we consider the introductory one: the definition of the judgement $\mathsf{allPos}(s)$, where s is a stream of natural numbers, which, intuitively, should hold when s positive, that is, contains only positive elements. We report here the inference system $\mathcal{I}^{>0}$ defining this predicate:

$$\frac{\mathsf{allPos}(s)}{\mathsf{allPos}(x{:}s)}\ x > 0$$

The specification $\mathcal{S}^{>0}$ is the set of judgements $\mathsf{allPos}(s)$, where s is rational, meaning that it has finitely many different substreams, and positive. Then, the completeness statement is the following:

If s is rational and positive, then $\mathcal{I}^{>0} \vdash_{\rho} \mathsf{allPos}(s)$.

To prove the result, let s be a rational stream containing only positive elements and set $X_s = \{\mathsf{allPos}(s') \mid s = x_1{:}\ldots{:}x_n{:}s'\}$. Clearly, X_s is finite, because s is rational, and $\mathsf{allPos}(s) \in X_s$, hence we have only to prove that it is consistent. Let $\mathsf{allPos}(s') \in X_s$, then $s' = x{:}s''$, thus $s = x_1{:}\ldots{:}x_n{:}x{:}s''$, and so $x > 0$, because it is an element of s, and $\mathsf{allPos}(s'') \in X_s$, by definition of X_s, and this proves that X_s is post-fixed. Therefore, by the regular coinduction principle we get the thesis.

Let us now focus on the soundness property. If we interpreted $\mathcal{I}$ inductively, we would prove soundness by induction on rules, but in the regular case this technique is not available, since it is unsound. However, in Theorem 8.12, we proved that $j \in \rho[\![\mathcal{I}]\!]$ if and only if $\emptyset \triangleright j$ is derivable in $\mathcal{I}^{\circlearrowleft}$, which is interpreted inductively. Therefore, we can exploit the induction principle associated with $\mathcal{I}^{\circlearrowleft}$ to prove soundness, as the following proposition states:

PROPOSITION 8.19 : Let $\mathcal{S} \subseteq \mathcal{U}$ be a set of judgements, then, if there is a family $(\mathcal{S}_H)_{H \in \wp_\omega(\mathcal{U})}$ such that $\mathcal{S}_H \subseteq \mathcal{U}$ and $\mathcal{S}_\emptyset \subseteq \mathcal{S}$, and, for all $H \in \wp_\omega(\mathcal{U})$,

- $H \subseteq \mathcal{S}_H$, and
- for all rules $\langle Pr, j\rangle \in \mathcal{I}$, if $Pr \subseteq \mathcal{S}_{H\cup\{j\}}$ then $j \in \mathcal{S}_H$,

then $\rho[\![\mathcal{I}]\!] \subseteq \mathcal{S}$.

Proof: By induction on $\mathcal{I}^{\circlearrowleft}$, we immediately get that $\mathcal{I}^{\circlearrowleft} \vdash_\mu H \triangleright j$ implies $j \in \mathcal{S}_H$. Therefore, if $j \in \rho[\![\mathcal{I}]\!]$, by Theorem 8.12, we have $\mathcal{I}^{\circlearrowleft} \vdash_\mu \emptyset \triangleright j$, hence $j \in \mathcal{S}_\emptyset \subseteq \mathcal{S}$. □

In other words, given a specification $\mathcal{S} \subseteq \mathcal{U}$, to prove soundness we have first to generalize the specification to a family of specifications, indexed over finite sets of judgements, in order to take into account circular hypotheses. Then, we reason by induction on rules in the equivalent inductive system (see Definition 8.11) and, since $\mathcal{S}_\emptyset \subseteq \mathcal{S}$, we get soundness.

EXAMPLE 8.20 : We illustrate the technique again on the definition of $\mathsf{allPos}(s)$. The soundness statement is the following:

> If $\mathcal{I}^{>0} \vdash_\rho \mathsf{allPos}(s)$, then s is rational and positive.

The first step is to generalize the specification to a family $(\mathcal{S}^{>0}_H)$, indexed over finite subsets of judgements H.

> $\mathsf{allPos}(s) \in \mathcal{S}^{>0}_H$ iff either s is rational and positive, or $s = x_1{:}\ldots{:}x_n{:}s'$ with $x_i > 0$, for all $i \in 1..n$, and $\mathsf{allPos}(s') \in H$.

It is easy to see that $\mathcal{S}^{>0}_\emptyset \subseteq \mathcal{S}^{>0}$ and, for all $H \in \wp_\omega(\mathcal{U})$, $H \subseteq \mathcal{S}^{>0}_H$, by definition of $\mathcal{S}^{>0}_H$. Hence, we have only to check that $(\mathcal{S}^{>0}_H)$ is closed with respect to the rule, as formulated in Proposition 8.19.

Let us assume $\mathsf{allPos}(s) \in \mathcal{S}^{>0}_{H'}$, with $H' = H \cup \{\mathsf{allPos}(x{:}s)\}$. We have the following cases:

- If s is rational and positive, this is true for $x{:}s$ as well, because $x > 0$ by hypothesis.
- If $s = x_1{:}\ldots{:}x_n{:}s'$ with $x_i > 0$, for all $i \in 1..n$, and $\mathsf{allPos}(s') \in H'$, then, if $\mathsf{allPos}(s') \in H$, since $x{:}s = x{:}x_1{:}\ldots{:}x_n{:}s'$ and $x > 0$, we have the thesis; if $s' = x{:}s$ then $x{:}s = x{:}x_1{:}\ldots{:}x_n{:}x{:}s$, thus it is rational and positive, as $x > 0$.

We now consider a more complex example: the definition of the distance in a graph (see page 180), proving it is sound and complete with respect to the expected meaning.

EXAMPLE 8.21 : For the reader's convenience, we report here the rules defining this judgement:

$$(\textsc{empty})\ \frac{}{\mathsf{dist}_G(v, v, 0)} \qquad (\textsc{adj})\ \frac{\mathsf{dist}_G(v_1, u, \delta_1) \quad \ldots \quad \mathsf{dist}_G(v_n, u, \delta_n) \quad v \neq u}{\mathsf{dist}_G(v, u, 1 + \min\{\delta_1, \ldots, \delta_n\}) \quad G(v) = \{v_1, \ldots, v_n\}}$$

We denote by $\mathcal{I}^{\mathsf{dist}}$ the above inference system. We recall for the reader's convenicence a few definitions we need in the proof. Let us assume a graph $G : V \to \wp(V)$. An edge in G is a pair $\langle v, u\rangle$ such that $u \in G(v)$, often written vu. We denote by E the set of edges in G. A path from v_0 to u_n in G is a non-empty finite sequence of nodes $\alpha = v_0 \dots v_n$ with $n \geq 0$, such that, for all $i \in 1..n$, $v_{i-1}v_i \in E$. The empty path starting from the node v to itself is the sequence v. If α is a path in G, then we denote by $\|\alpha\|$ the length of α, that is, the number of edges in α, and we write $v \in \alpha$ when the node v occurs in α, that is, the path α traverses v. The distance from a node v to a node u, denoted by $\delta(v, u)$, is the least length of a path from v to u, that is, $\delta(v, u) = \inf\{\|\alpha\| \mid \alpha \text{ is a path from } v \text{ to } u\}$, hence, if there is no path from v to u, $\delta(v, u) = \inf \emptyset = \infty$. We say a path $\alpha = v_0 \dots v_n$ is *simple* if it visits every node at most once, that is, $v_i = v_j$ implies $i = j$, for all $i, j \in 0..n$. Note that the empty path is trivially simple. It is also important to note that $\delta(v, u)$ is the least length of a simple path from v to u. Then, the specification $\mathcal{S}^{\mathsf{dist}}$ is the set of judgements $\mathsf{dist}_G(v, u, \delta)$ with $\delta = \delta(v, u)$.

We can now state that the definition of $\mathsf{dist}_G(v, u, \delta)$ is sound and complete with respect to the specification $\mathcal{S}^{\mathsf{dist}}$.

$$\mathcal{I}^{\mathsf{dist}} \vdash_{\rho} \mathsf{dist}_G(v, u, \delta) \text{ iff } \delta = \delta(v, u).$$

COMPLETENESS PROOF The proof is by regular coinduction. Let us consider a judgement $\mathsf{dist}_G(v, u, \delta(v, u))$. Let $R_v \subseteq V$ be the set of nodes reachable from v, and define $X_v = \{\mathsf{dist}_G(v', u, \delta(v', u)) \mid v' \in R_v\}$, which is clearly finite and $\mathsf{dist}_G(v, u, \delta(v, u)) \in X_v$, because v is reachable from itself. Hence, we have only to prove that X_v is post-fixed. Let $v' \in R_v$, then we have to find a rule with conclusion $\mathsf{dist}_G(v', u, \delta_{v',u})$ and whose premises are in X_v. We have two cases:

- If $v' = u$, then $\delta(v', u) = 0$ and so we have the thesis by rule (EMPTY).
- If $v' \neq u$, then we have $\delta(v', u) = 1 + \inf\{\delta(v'', u) \mid v'' \in G(v')\}$, hence, since $G(v') \subseteq R_v$, all the premises $\mathsf{dist}_G(v'', u, \delta(v'', u))$, for $v'' \in G(v')$, belong to X_v, as needed.

SOUNDNESS PROOF To apply Proposition 8.19, we generalize the specification $\mathcal{S}^{\mathsf{dist}}$ to a family $(\mathcal{S}^{\mathsf{dist}}_H)$, indexed over finite sets of judgements, defined below.

> ($\star$) $\mathsf{dist}_G(v, u, \delta) \in \mathcal{S}^{\mathsf{dist}}_H$ iff there is a set of paths P and a function $f : P \to \mathbb{N} \cup \{\infty\}$ such that
>
> 1. for all $\alpha \in P$, either α goes from v to u and $f(\alpha) = 0$, or α goes from v to v' and $\mathsf{dist}_G(v', u, f(\alpha)) \in H$;
> 2. for each simple path β from v to u, there is $\alpha \in P$ such that $\beta = \alpha\beta'$;
> 3. $\delta = \inf\{\|\alpha\| + f(\alpha) \mid \alpha \in P\}$.

First, we have to check that $\mathcal{S}^{\mathsf{dist}}_{\emptyset} \subseteq \mathcal{S}^{\mathsf{dist}}$. Let $\mathsf{dist}_G(v, u, \delta) \in \mathcal{S}^{\mathsf{dist}}_{\emptyset}$, then, by Item 3 of $(\star)$, $\delta = \inf\{\|\alpha\| + f(\alpha) \mid \alpha \in P\}$, for some set of paths P and function $f : P \to \mathbb{N} \cup \{\infty\}$. Since H is empty, by Item 1 of $(\star)$, we have that, for all $\alpha \in P$, α is a path from v to u and $f(\alpha) = 0$, hence $\delta(v, u) \leq \|\alpha\| + f(\alpha)$ for all $\alpha \in P$ and so $\delta(v, u) \leq \delta$. To prove the other inequality, let β be a simple path from v to u, then, by Item 2 of $(\star)$, there is $\alpha \in P$ such that $\beta = \alpha\beta'$, but, by Item 1 of $(\star)$, α goes from v to u and $f(\alpha) = 0$, hence $\beta = \alpha$, because it cannot traverse twice u; thus we have $\delta \leq \|\beta\| = \|\alpha\| + f(\alpha)$, for any simple path β, and so $\delta \leq \delta(v, u)$.

The fact that $H \subseteq \mathcal{S}^{\mathsf{dist}}_H$ is immediate because, if $\mathsf{dist}_G(v, u, \delta) \in H$, then, to get the thesis, it is enough to take as P the set containing only the empty path with $f(v) = \delta$, which trivially satisfies all conditions in $(\star)$.

Then, we have only to check that $(\mathcal{S}^{\mathsf{dist}}_H)$ is closed with respect to the rules (EMPTY) and (ADJ), as formulated in Proposition 8.19.

Case: (EMPTY) If $v = u$ and $\delta = 0$, then it is enough to take as P the set containing only the empty path, with $f(v) = 0$.

Case: (ADJ) We have $v \neq u$, $G(v) = \{v_1, \ldots, v_n\}$ and $\mathsf{dist}_G(v_i, u, \delta_i) \in \mathcal{S}^{\mathsf{dist}}_{H'}$ with $H' = H \cup \{\mathsf{dist}_G(v, u, \delta)\}$, for all $i \in 1..n$. If $n = 0$, then $G(v)$ is empty, $\delta = \inf \emptyset = \infty$ and there is no path from v to u. Hence, the thesis follows by taking $P = \emptyset$.

Then, let us assume $n \geq 1$. By hypothesis, $\delta = 1 + \inf\{\delta_1, \ldots, \delta_n\} = 1 + \delta_k$, for some $k \in 1..n$, since we are considering rule (ADJ). Since $\mathsf{dist}_G(v_i, u, \delta_i) \in \mathcal{S}^{\mathsf{dist}}_{H'}$, for all $i \in 1..n$, there are P_i and $f_i : P_i \to \mathbb{N} \cup \{\infty\}$ satisfying $(\star)$, in particular, by Item 3, $\delta_i = \inf\{\|\alpha\| + f_i(\alpha) \mid \alpha \in P_i\}$. We define P as the set of paths $v\alpha$ with $\alpha \in P_i$ such that, if α ends in v, then $f_i(\alpha) \neq \delta$, and $f : P \to \mathbb{N} \cup \{\infty\}$ is defined by $f(v\alpha) = f_i(\alpha)$ when $\alpha \in P_i$. Clearly, P satisfies Item 1 of $(\star)$ with respect to H. To check that Item 2 holds, let β be a simple path from v to u, then $\beta = vv_i\beta'$, for some $i \in 1..n$. Hence, $v_i\beta'$ is a simple path from v_i to u and $v \notin v_i\beta'$, thus, by Item 2 of $(\star)$ applied to P_i, there is $\alpha' \in P_i$ such that $v_i\beta' = \alpha'\gamma$, and $v \notin \alpha'$, because $v \notin v_i\beta'$. Therefore, $v\alpha' \in P$ and $v\alpha'\gamma = vv_i\beta' = \beta$, as needed.

We now prove Item 3 of $(\star)$, that is, $\delta = \inf\{\|\alpha\| + f(\alpha) \mid \alpha \in P\}$. Let $\alpha = vv_i\alpha' \in P$, for some $i \in 1..n$, then $v_i\alpha' \in P_i$, hence, $\delta_k \leq \delta_i \leq \|v_i\alpha'\| + f_i(v_i\alpha')$, thus $\delta = 1 + \delta_k \leq 1 + \|v_i\alpha'\| + f_i(v_i\alpha') = \|\alpha\| + f(\alpha)$, which implies $\delta \leq \inf\{\|\alpha\| + f(\alpha) \mid \alpha \in P\}$. To conclude, we have to prove the other inequality, hence we distinguish the following cases:

- if $\delta_k = \infty$, then $\delta = \infty$ and this proves the thesis, since $\infty \geq x$ for all $x \in \mathbb{N} \cup \{\infty\}$;
- otherwise, $\delta_k = \|\alpha'\| + f_k(\alpha')$, for some $\alpha' \in P_k$. If α' ends in v and $f_k(\alpha') = \delta$, then $\delta_k = \|\alpha'\| + \delta = \|\alpha'\| + 1 + \delta_k$, which implies $\delta_k = \infty$ that is absurd. Otherwise, $v\alpha' \in P$ and $f(v\alpha') = f_k(\alpha')$, thus $\inf\{\|\alpha\| + f(\alpha) \mid \alpha \in P\} \leq \|v\alpha'\| + f(v\alpha') = 1 + \|\alpha'\| + f_k(\alpha') = 1 + \delta_k = \delta$, as needed.

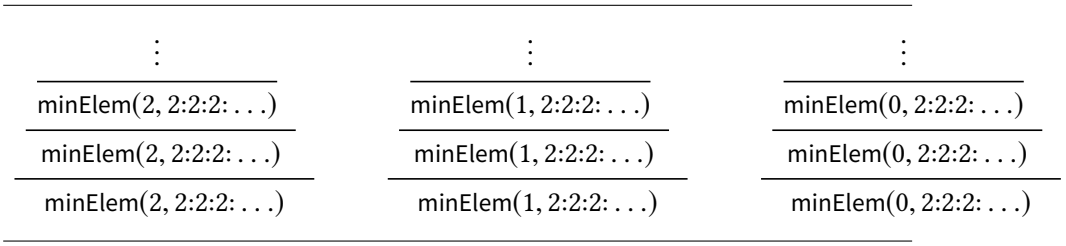


FIGURE 8.1 Some infinite regular derivation for the judgement $\mathsf{minElem}(x, s)$.

8.6 Flexible regular coinduction

Infinite derivations are a very powerful tool, which make it possible to deal with a variety of situations that cannot be handled just by finite ones. However, in some cases, as widely discussed in Chapter 3, they have an unexpected behaviour, allowing the derivation of intuitively incorrect judgements. Not surprisingly, the same issue affects also regular derivations. Let us explain this by an example, which is a slight variation of one in Chapter 3. Consider the following rules, defining the judgement $\mathsf{minElem}(x, l)$, where x is an integer and s is a rational stream, stating that x is the minimum of the stream s.

$$\frac{\mathsf{minElem}(y, s)}{\mathsf{minElem}(z, x{:}s)}\ z = \min\{x, y\}$$

In Figure 8.1 we report three infinite regular derivations, thus valid for the regular interpretation of the above rules, where, however, only the first one is intuitively correct: judgements $\mathsf{minElem}(0, 2{:}2{:}\ldots)$ and $\mathsf{minElem}(1, 2{:}2{:}\ldots)$ should not be derivable, as 0 and 1 do not belong to the stream.

Inference systems with corules, introduced in Chapter 3, have been designed precisely to address this issue for the coinductive interpretation, where arbitrary infinite derivations are allowed. Indeed, corules allow refinement of the coinductive interpretation, by filtering out some, undesired, infinite derivations.

In this section, we show that the results previously given for regular coinduction smoothly extend to generalised inference systems. The technical development in the following is partly repetitive; this could have been avoided by presenting the results in the generalized framework since the beginning. However, to have separation of concerns, we preferred to first give a presentation using only standard notions, limiting to this section the non-standard ones.

We start by defining the regular interpretation of an inference system with corules $\langle \mathcal{I}, .\mathcal{I}_{\mathsf{co}} \rangle$. Recall that, given an inference system $\mathcal{I}$ and a set of judgements $X \subseteq \mathcal{U}$, $\mathcal{I}_{|X}$ is the subset of $\mathcal{I}$ containing only rules with conclusion in X.

DEFINITION 8.22 : Let $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$ be an inference system with corules. The *regular interpretation* $\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$ of $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$ is defined by $\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] = \rho[\![\mathcal{I}_{|\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]}]\!]$.

As we will see later in this section (Corollary 8.28), in proof-theoretic terms this is equivalent to say that $\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$ is the set of judgements with a regular

proof tree in $\mathcal{I}$, whose nodes all have a finite proof tree in $\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}$. In this way, we can filter out some, undesired, regular derivations. In the following, we will write $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle \vdash_\rho j$ for $j \in \rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$.

Coming back to the example, using corules, we can provide a correct definition of the judgement $\mathsf{minElem}(x, s)$ as follows:

$$\frac{\mathsf{minElem}(y, s)}{\mathsf{minElem}(z, x{:}s)}\ z = \min\{x, y\} \qquad \frac{}{\mathsf{minElem}(x, x{:}s)}$$

The additional constraint, imposed by the coaxiom, allows us to build regular infinite derivation using only judgements $\mathsf{minElem}(x, s)$ where x belongs to s; thus filtering out the second and third incorrect proof trees in Figure 8.1, since they involve judgements with no finite derivation using also the coaxiom.

All the results discussed so far for the regular interpretation can be smoothly extended to the regular interpretation of an inference system with corules. We will now develop all the tecnical machinery needed for this, adapting constructions in Chapter 3 to the regular case.

8.6.1 Bounded rational fixed point

To construct such a fixed point, we come back to the lattice-theoretic setting of Section 8.2. Let us assume an algebraic lattice $\langle L, \sqsubseteq \rangle$.

Let $F, G : L \to L$ be two functions, we write $F \sqcup G$ for the pointwise join of F and G, and, for all $z \in L$, $F_{\sqcap z}$ for the function defined by $F_{\sqcap z}(x) = F(x) \sqcap z$. It is easy to see that, if F and G are monotone, then $F \sqcup G$ is monotone as well, hence, by the Tarski theorem, it has a least fixed point $\mu(F \sqcup G)$. It is also easy to check that, if $z \in L$ is a pre-fixed point of $F \sqcup G$, then it is a pre-fixed point of F as well, because $F(z) \sqsubseteq F(z) \sqcup G(z) \sqsubseteq z$; this will be crucial for the following construction, as it was for the analogous construction of Section 3.2.1.

We can now define the bounded rational fixed point:

DEFINITION 8.23 : Let $F : L \to L$ be finitary and $G : L \to L$ be monotone. The *rational fixed point bounded by* G, $\rho[F, G]$ is defined by

$$\rho[F, G] = \rho F_{\sqcap \mu(F \sqcup G)}$$

In other words, $\rho[F, G]$ is the least upper bound of all compact elements *below* the least fixed point of $F \sqcup G$, that is,

$$\rho[F, G] = \bigsqcup \{x \in C(L) \mid x \sqsubseteq F(x),\ x \sqsubseteq \mu(F \sqcup G)\}$$

To see that $\rho[F, G]$ is well-defined, that is, it is indeed a fixed point of F, we have the following propositions:

PROPOSITION 8.24 : If $F : L \to L$ is finitary, then, for all $z \in L$, $F_{\sqcap z}$ is finitary as well.

Proof: Let $D \subseteq L$ be a directed set. Since F is finitary, it is monotone, hence $F_{\sqcap z}$ is monotone as well, therefore we get $\bigsqcup(F_!(D) \sqcap z) \sqsubseteq F(\bigsqcup D) \sqcap z$. To prove the other inequality, it is enough to show that, for any compact element $y \sqsubseteq F(\bigsqcup D) \sqcap z$, $y \sqsubseteq \bigsqcup(F_!(D) \sqcap z)$, because the lattice is algebraic.

Since F is finitary, we have $F(\bigsqcup D) = \bigsqcup F_!(D)$. We know that $y \sqsubseteq z$ and $y \sqsubseteq F(\bigsqcup D) = \bigsqcup F_!(D)$ and, since y is compact, there is a finite subset $W \subseteq D$ such that $y \sqsubseteq \bigsqcup F_!(W)$. Since D is directed and W is finite, there is $w \in D$ such that $\bigsqcup W \sqsubseteq w$, hence $\bigsqcup F_!(W) \sqsubseteq F(w) \sqsubseteq \bigsqcup F_!(D)$, because F is monotone and $w \in D$. Therefore, we get $y \sqsubseteq F(w) \sqcap z \sqsubseteq \bigsqcup(F_!(D) \sqcap z)$, as needed. □

PROPOSITION 8.25 : Let $F : L \to L$ be finitary and $G : L \to L$ be monotone, then $\rho[F, G]$ is a fixed point of F.

Proof: Set $z = \mu(F \sqcup G)$ and note that $F(z) \sqsubseteq F(z) \sqcup G(z) = z$, as z is a fixed point of $F \sqcup G$. By Proposition 8.24, $F_{\sqcap z}$ is finitary, hence, by Definition 8.23 and Theorem 8.4 we have $\rho[F, G] = F(\rho[F, G]) \sqcap z$, and from this we derive $\rho[F, G] \sqsubseteq z$ and $F(\rho[F, G]) \sqsubseteq F(z) \sqsubseteq z$. Therefore, we get $\rho[F, G] = F(\rho[F, G]) \sqcap z = F(\rho[F, G])$, as needed. □

In Chapter 3 we showed that the least and the greatest fixed point are instances of the bounded fixed point (cf. Proposition 3.8). Analogously, we show that the least and the rational fixed point are instances of the bounded rational fixed point, that is, they can be recovered for specific choices of G. In the following, for all $z \in L$, we write $K_z : L \to L$ for the constant function, that is, $K_z(x) = z$, for all $x \in L$.

PROPOSITION 8.26 : Let $F : L \to L$ be a finitary function, then the following hold:

1. $\mu F = \rho[F, K_\bot]$, and
2. $\rho F = \rho[F, K_\top]$.

Proof: To prove 1, note that $\mu F \sqsubseteq \rho[F, K_\bot]$, as $\rho[F, K_\bot]$ is a pre-fixed point, and $\rho[F, K_\bot] \sqsubseteq \mu F$, as $\mu(F \sqcup K_\bot) = \mu F$ and $\rho[F, K_\bot] = F(\rho[F, K_\bot]) \sqcap \mu F \sqsubseteq \mu F$. To prove 2, note that $\mu(F \sqcup K_\top) = \top$, hence we have $F_{\sqcap \mu(F \sqcup K_\top)} = F$, thus $\rho[F, K_\top] = \rho F$, as needed. □

8.6.2 Fixed point semantics

Let $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$ be an inference system with corules where $\mathcal{I}$ is finitary. We have two goals: first we want to justify the proof-theoretic characterisation provided at the beginning of this section and, second, we want to prove that the rational interpretation generated by corules is indeed an interpretation of the inference system.

To get the proof-theoretic characterisation, it is enough to observe the following property:

PROPOSITION 8.27 : Let $X \subseteq \mathcal{U}$, then $\tau \in \rho T_{\mathcal{I}_{\restriction X}}$ iff $\tau \in \rho T_{\mathcal{I}}$ and, for all $\alpha \in \mathsf{N}(\tau)$, $\tau(\alpha) \in X$.

Proof: By definition we have $\mathcal{I}_{|X} \subseteq \mathcal{I}$, hence $\rho T_{\mathcal{I}_{|X}} \subseteq \rho T_{\mathcal{I}}$, all rules in $\mathcal{I}_{|X}$ have conclusion in X, and, by Lemma 8.8, $\rho T_{\mathcal{I}}$ is the set of regular proof trees in $\mathcal{I}$. Then, the thesis is immediate since, by definition, all nodes of a proof tree are labelled by the conclusion of some rule. □

Recall that we have described $\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$ in proof-theoretic terms as the set of judgements having a regular proof tree in $\mathcal{I}$, whose nodes all have a finite proof tree in $\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}$. Formally, we have the following corollary:

COROLLARY 8.28 : $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle \vdash_\rho j$ iff there is $\tau \in \rho T_{\mathcal{I}}$ such that $\mathsf{r}(\tau) = j$ and, for all $\alpha \in \mathsf{N}(\tau)$, $\tau(\alpha) \in \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$.

Proof: Set $X = \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$. From Theorem 8.9 and Definition 8.22 we get $\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] = \rho[\![\mathcal{I}_{|X}]\!] = \mathsf{r}_!(\rho T_{(\mathcal{I}_{|X})}) = \rho F_{(\mathcal{I}_{|X})}$. Applying Proposition 8.27 with $X = \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$, we get the thesis. □

Towards the second goal, we show that the regular interpretation of $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$ coincides with the rational fixed point of $F_{\mathcal{I}}$ bounded by $F_{\mathcal{I}_{\mathsf{co}}}$ (see Definition 8.23), which is an immediate consequence of the following proposition:

PROPOSITION 8.29 : $\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] = \rho[F_{\mathcal{I}}, F_{\mathcal{I}_{\mathsf{co}}}]$.

Proof: By Definition 8.22 and Theorem 8.9, we know that $\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] = \rho F_{(\mathcal{I}_{|\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]})}$. By Definition 8.23, we have $\rho[F_{\mathcal{I}}, F_{\mathcal{I}_{\mathsf{co}}}] = \rho(F_{\mathcal{I}})_{\sqcap \mu(F_{\mathcal{I}} \cup F_{\mathcal{I}_{\mathsf{co}}})}$ and, by definition of the inference operator, we have $F_{\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}} = F_{\mathcal{I}} \cup F_{\mathcal{I}_{\mathsf{co}}}$, hence $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!] = \mu(F_{\mathcal{I}} \cup F_{\mathcal{I}_{\mathsf{co}}})$ and $(F_{\mathcal{I}})_{\sqcap \mu(F_{\mathcal{I}} \cup F_{\mathcal{I}_{\mathsf{co}}})} = (F_{\mathcal{I}})_{\sqcap \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]}$. Therefore, by Proposition 3.12, we have $F_{(\mathcal{I}_{|\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]})} = (F_{\mathcal{I}})_{\sqcap \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]}$, which implies the thesis. □

Then, this proposition, together with Proposition 8.25, in particular ensures that $\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$ is indeed a fixed point of $F_{\mathcal{I}}$, that is, an $\mathcal{I}$-interpretation (cf. Definition 2.2).

An important property of inference systems with corules is that standard interpretations (the inductive and the coinductive one) are particular cases (cf. Corollary 3.15). Analogously, the inductive and the regular interpretations are particular cases of the regular interpretation generated by corules. Recall that $\mathcal{I}_{\mathcal{U}}$ denotes the inference system consisting of one axiom for each $j \in \mathcal{U}$. We have the following proposition:

PROPOSITION 8.30 : Let $\mathcal{I}$ be an inference system, then $\mu[\![\mathcal{I}]\!] = \rho[\![\mathcal{I}, \emptyset]\!]$ and $\rho[\![\mathcal{I}]\!] = \rho[\![\mathcal{I}, \mathcal{I}_{\mathcal{U}}]\!]$.

Proof: It follows from Proposition 8.26, because $\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] = \rho[F_{\mathcal{I}}, F_{\mathcal{I}_{\mathsf{co}}}]$, by Proposition 8.29, and we have $F_{\mathcal{I}_{\mathcal{U}}}(X) = \mathcal{U}$ and $F_{\emptyset}(X) = \emptyset$, for all $X \subseteq \mathcal{U}$. □

In other words, when the set of corules is empty, we allow only rules with conclusion in $\mu[\![\mathcal{I} \cup \emptyset]\!] = \mu[\![\mathcal{I}]\!]$, hence we cannot derive anything outside $\mu[\![\mathcal{I}]\!]$,

and, on the other hand, when the set of corules is $\mathcal{I}_{\mathcal{U}}$, we do not remove any rule, because $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathcal{U}}]\!] = \mathcal{U}$, thus we get exactly the regular interpretation of $\mathcal{I}$.

8.6.3 Cycle detection for corules

As the standard regular interpretation, also the regular interpretation of an inference system with corules has a sound and complete algorithm to find a derivation for a judgment, if any.

Let us assume an inference system with corules $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$. Since its regular interpretation is defined as the regular interpretation of $\mathcal{I}_{|\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]}$, which is the inference system obtained from $\mathcal{I}$ by keeping only rules with conclusion in $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$, we could get an inductive characterisation of $\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$ by applying the construction in Definition 8.11 to the inference system $\mathcal{I}_{|\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]}$. This provides us with a sound and complete algorithm to find a derivation for a judgement which belongs to $\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$, which works the same way as the one introduced in Section 8.4, but, in addition, each time we apply the rule (UNFOLD) with $\langle Pr, j \rangle \in \mathcal{I}$, we have to check that $j \in \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$. However, we will see that this additional check is necessary only to apply circular hypotheses, thus defining a cleaner procedure.

To this end we construct the inference system $\mathcal{I}^{\circlearrowleft \mathcal{I}_{\mathsf{co}}}$ as follows:

DEFINITION 8.31 : The inference system $\mathcal{I}^{\circlearrowleft \mathcal{I}_{\mathsf{co}}}$ consists of the following rules:

$$\text{(B-HP)}\ \frac{j \in H \quad j \in \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]}{H \triangleright j}$$

$$\text{(B-UNFOLD)}\ \frac{H \cup \{j\} \triangleright j_1 \quad \dots \quad H \cup \{j\} \triangleright j_n}{H \triangleright j} \quad \langle \{j_1, \dots, j_n\}, j \rangle \in \mathcal{I}$$

This definition is basically the same as Definition 8.11, except for the additional side condition in rule (B-HP) $j \in \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$, which enforces the additional check. We have the following fundamental properties:

PROPOSITION 8.32 : If $\mathcal{I}^{\circlearrowleft \mathcal{I}_{\mathsf{co}}} \vdash_{\mu} H \triangleright j$ then $j \in \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$.

Proof: By induction on rules of $\mathcal{I}^{\circlearrowleft \mathcal{I}_{\mathsf{co}}}$: the case for rule (B-HP) is trivial, for the rule (B-UNFOLD), by Definition 8.31, we have a rule $\langle \{j_1, \dots, j_n\}, j \rangle \in \mathcal{I}$ and, by induction hypothesis, we know that $j_k \in \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$, for all $k \in 1..n$, hence $j \in \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$, as $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$ is closed with respect to $\mathcal{I}$. □

PROPOSITION 8.33 : If $H \subseteq \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$, then $\mathcal{I}^{\circlearrowleft \mathcal{I}_{\mathsf{co}}} \vdash_{\mu} H \triangleright j$ iff $\mathcal{I}^{\circlearrowleft}_{|\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]} \vdash_{\mu} H \triangleright j$.

Proof: The proof of the left-to-right implication is by induction on rules in $\mathcal{I}^{\circlearrowleft \mathcal{I}_{\mathsf{co}}}$.

Case: (B-HP) By hypothesis $j \in \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$, then the thesis follows by rule (HP).

Case: *(B-UNFOLD)* By Definition 8.31, we have a rule $\langle \{j_1, \ldots, j_n\}, j\rangle \in \mathcal{I}$ and by Proposition 8.32 we have $j \in \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$, hence $H \cup \{j\} \subseteq \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$ and $\langle \{j_1, \ldots, j_n\}, j\rangle \in \mathcal{I}_{|\mu[\![\mathcal{I}\cup\mathcal{I}_{\mathsf{co}}]\!]}$. Therefore, by induction hypothesis, we get $\mathcal{I}^{\circlearrowleft}_{|\mu[\![\mathcal{I}\cup\mathcal{I}_{\mathsf{co}}]\!]} \vdash_\mu H \cup \{j\} \triangleright j_k$, for all $k \in 1..n$, then the thesis follows by rule (UNFOLD).

The proof of the right-to-left implication is by induction on rules in $\mathcal{I}^{\circlearrowleft}_{|\mu[\![\mathcal{I}\cup\mathcal{I}_{\mathsf{co}}]\!]}$.

Case: *(HP)* Immediate by rule (B-HP), as $j \in H \subseteq \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$.

Case: *(UNFOLD)* By Definition 8.11, we have a rule $\langle \{j_1, \ldots, j_n\}, j\rangle \in \mathcal{I}_{|\mu[\![\mathcal{I}\cup\mathcal{I}_{\mathsf{co}}]\!]} \subseteq \mathcal{I}$, hence $j \in \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$, and so $H \cup \{j\} \subseteq \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$. Therefore, by induction hypothesis, we get $\mathcal{I}^{\circlearrowleft \mathcal{I}_{\mathsf{co}}} \vdash_\mu H \cup \{j\} \triangleright j_k$, for all $k \in 1..n$, then the thesis follows by rule (B-UNFOLD).

□

Then, we get the following result, proving that the inductive characterisation is correct, that is, sound and complete, with respect to the regular interpretation of $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle$.

COROLLARY 8.34 : $\mathcal{I}^{\circlearrowleft \mathcal{I}_{\mathsf{co}}} \vdash_\mu \emptyset \triangleright j$ iff $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}}\rangle \vdash_\rho j$.

Proof: It is immediate by Proposition 8.33 and Theorem 8.12. □

The resulting algorithm behaves as follows: we start from a judgement j with an empty set of circular hypotheses, then we try to build a regular derivation for j using rules in $\mathcal{I}$, exactly the same way as for standard regular coinduction; but, this time, when we find a cycle, say for a judgement j', we trigger another procedure, which looks for a finite derivation for j' in $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$.

8.6.4 Flexible regular reasoning

We now adapt proof techniques presented in Section 8.5 to this generalised setting. For completeness proofs, in Section 3.4, the standard coinduction principle is extended to generalised inference systems, by adding an additional constraint, which takes into account corules (cf. Proposition 3.27). The regular coinduction principle (cf. Proposition 8.17) can be smoothly extended to this generalised setting following the same strategy, as expressed in the next proposition. We call the resulting principle the *bounded regular coinduction principle.*

PROPOSITION 8.35 (Bounded regular coinduction): Let $\mathcal{S} \subseteq \mathcal{U}$ be a set of judgements. If, for all $j \in \mathcal{S}$, there is a finite set $X \subseteq \mathcal{U}$ such that

BOUNDEDNESS $j \in X \subseteq \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$, and

CONSISTENCY $X \subseteq F_{\mathcal{I}}(X)$,

then, $\mathcal{S} \subseteq \rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$.

This proposition immediately follows from Proposition 8.5, as $\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$ is a rational fixed point by Proposition 8.29 and Definition 8.23. The additional constraint $\mathcal{S} \subseteq \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$, named *boundedness*, reflects the fact that, using corules, we are only allowed to build proof trees using judgements in $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$. Note that, when $\mathcal{I}_{\mathsf{co}} = \mathcal{I}_{\mathcal{U}}$, thus $\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] = \rho[\![\mathcal{I}]\!]$, the additional constraint is trivially true, because it requires $\mathcal{S} \subseteq \mathcal{U}$, hence we recover the regular coinduction principle in Proposition 8.17.

EXAMPLE 8.36 : We illustrate this technique on our running example: the definition of $\mathsf{minElem}(x, s)$, which should hold when x is the minimum of the rational stream of integers s. We denote by $\langle \mathcal{I}^{\mathsf{min}}, \mathcal{I}^{\mathsf{min}}_{\mathsf{co}} \rangle$ the inference system with corules defining the judgement $\mathsf{minElem}(x, s)$, and by $\mathcal{S}^{\mathsf{min}}$ the set of judgements $\mathsf{minElem}(x, s)$ where x is indeed the minimum of s. We prove, using Proposition 8.35, the following statement:

$$\text{if } \mathsf{minElem}(x, s) \in \mathcal{S}^{\mathsf{min}} \text{ then } \langle \mathcal{I}^{\mathsf{min}}, \mathcal{I}^{\mathsf{min}}_{\mathsf{co}} \rangle \vdash_{\rho} \mathsf{minElem}(x, s).$$

Let $\mathsf{minElem}(x, s) \in \mathcal{S}^{\mathsf{min}}$ and define X as the set of judgements $\mathsf{minElem}(z, r) \in \mathcal{S}^{\mathsf{min}}$ such that r is a substream of s. Trivially $\mathsf{minElem}(x, s) \in X$ and, since s is rational, it has finitely many different substreams, hence X is finite. The boundedness condition, that is, if $\mathsf{minElem}(z, r) \in X$ then it has a finite proof tree using also the coaxioms, is easy to check, because, if y is the minimum of r, then y occurs somewhere in r, hence we can prove the thesis by induction on the least position of y in r. In order to check that X is consistent, consider $\mathsf{minElem}(z, r) \in X$, with $r = y{:}r'$. Since z is the minimum of r and r' is a substream of r, z is a lower bound of r', thus it has a minimum, say y', and so $\mathsf{minElem}(y', r') \in X$. To conclude, we have to show that $z = \min\{y, y'\}$. The inequality $z \leq \min\{y, y'\}$ is trivial, for the other inequality, since z belongs to r, we have two cases: if $z = y$, then $\min\{y, y'\} \leq z$, otherwise z belongs to r' and so $y' \leq z$, thus $\min\{y, y'\} \leq z$.

Differently from the standard coinductive interpretation, for the regular interpretation we have also defined a proof technique to show soundness (Proposition 8.19). Such a technique relies on the inductive characterisation of the regular interpretation. As also the regular interpretation of an inference system with corules has an inductive characterisation (Corollary 8.34), we can provide a proof technique to show soundness also in this generalised setting, which smoothly extends the one of standard regular coinduction.

PROPOSITION 8.37 : Let $\mathcal{S} \subseteq \mathcal{U}$ be a set of judgements, then, if there is a family $(\mathcal{S}_H)_{H \in \wp_\omega(\mathcal{U})}$ such that $\mathcal{S}_H \subseteq \mathcal{U}$ and $\mathcal{S}_\emptyset \subseteq \mathcal{S}$, and, for all $H \in \wp_\omega(\mathcal{U})$,

- $H \cap \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!] \subseteq \mathcal{S}_H$, and
- for all rules $\langle Pr, j \rangle \in \mathcal{I}$, if $Pr \subseteq \mathcal{S}_{H \cup \{j\}}$ then $j \in \mathcal{S}_H$,

then $\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] \subseteq \mathcal{S}$.

Proof: By a straightforward induction on rules in $\mathcal{I}^{\cup\mathcal{I}_{\text{co}}}$, we get that if $\mathcal{I}^{\cup\mathcal{I}_{\text{co}}} \vdash_\mu H \triangleright j$, then $j \in \mathcal{S}_H$; thus, the thesis follows from Corollary 8.34. □

Again, this proof principle is almost the same as Proposition 8.19, but with an additional constraint, this time on sets of circular hypotheses, which takes into account corules.

EXAMPLE 8.38 : We illustrate this technique proving that the definition of $\mathsf{minElem}(x, s)$ is sound, that is,

$$\text{if } \langle \mathcal{I}^{\mathsf{min}}, \mathcal{I}^{\mathsf{min}}_{\mathsf{co}} \rangle \vdash_\rho \mathsf{minElem}(x, s), \text{ then } x \text{ is the minimum of } s.$$

First of all, we note that, if $\mathsf{minElem}(x, s)$ has a finite proof tree using also the coaxiom, then x belongs to s (it can be easily proved by induction on rules in $\mathcal{I}^{\mathsf{min}} \cup \mathcal{I}^{\mathsf{min}}_{\mathsf{co}}$). Then, we define $\mathcal{S}^{\mathsf{min}}_H$ as follows: $\mathsf{minElem}(x, s) \in \mathcal{S}^{\mathsf{min}}_H$ iff x is the minimum of s or $s = x_1{:}\ldots{:}x_n{:}r$, $\mathsf{minElem}(y, r) \in H$, $\mathcal{I}^{\mathsf{min}} \cup \mathcal{I}^{\mathsf{min}}_{\mathsf{co}} \vdash_\mu \mathsf{minElem}(y, r)$ and $x = \min\{x_1, \ldots, x_n, y\}$. We have trivially that $\mathcal{S}^{\mathsf{min}}_\emptyset \subseteq \mathcal{S}^{\mathsf{min}}$.

Assume a finite set of judgements H. Clearly, if $\mathsf{minElem}(x, s) \in H$ has a finite proof tree using also the coaxiom, then $\mathsf{minElem}(x, s) \in \mathcal{S}^{\mathsf{min}}_H$. Now, suppose $s = x{:}r$, $H' = H \cup \{\mathsf{minElem}(z, s)\}$, $\mathsf{minElem}(y, r) \in \mathcal{S}^{\mathsf{min}}_{H'}$ and $z = \min\{x, y\}$, then we have two cases:

- if y is the minimum of r, then z is the minimum of $s = y{:}r$, hence $\mathsf{minElem}(z, s) \in \mathcal{S}^{\mathsf{min}}_H$;
- if $r = x_1{:}\ldots{:}x_n{:}r'$, $\mathsf{minElem}(y', r') \in H'$, $\mathsf{minElem}(y', r')$ has a finite proof tree using also the coaxiom and $y = \min\{x_1, \ldots, x_n, y'\}$, then $s = x{:}r = x{:}x_1{:}\ldots{:}x_n{:}r'$ and $z = \min\{x, x_1, \ldots, x_n, y'\}$. We distinguish two subcases:
  - if $\mathsf{minElem}(y', r') \in H$, then $\mathsf{minElem}(z, s) \in \mathcal{S}^{\mathsf{min}}_H$ by definition, and
  - if $y' = z$ and $r' = s$, then $s = x{:}x_1{:}\ldots{:}x_n{:}s$ and $\mathsf{minElem}(z, s)$ has a finite proof tree using also the coaxiom, thus z belongs to s and $z = \min\{x, x_1, \ldots, x_n, z\}$, that is, z is the minimum of s.

We now consider a more involved example, which is the restriction to the rational case of the example described in Section 3.5.1.

EXAMPLE 8.39 : It is well-known that real numbers in $[0, 1]$ can be represented as, not necessarily rational, streams of digits in some basis. Let $\mathbb{N}_{>0}$ be the set of positive natural numbers and assume a basis $b \in \mathbb{N}_{>0}$. A digit d is a natural number in $0..b-1$, then, given a stream $r = (d_i)_{i \in \mathbb{N}_{>0}}$ of digits, the series $\sum_{i=1}^{\infty} d_i b^{-i}$ converges and its limit is the real number represented by r and denoted by $[\![r]\!]$. It is also well-known that every real number $x \in [0, 1]$ has at most two different representations as a stream, for instance, with $b = 10$, the number $1/2$ can be represented as either $5{:}\overline{0}$ or $4{:}\overline{9}$, where, for any digit d, $\overline{d}$ is the stream $d{:}d{:}d{:}\ldots$.

Consider the following inference system with corules $\langle \mathcal{I}^{\mathsf{add}}, \mathcal{I}^{\mathsf{add}}_{\mathsf{co}} \rangle$, defining the judgement $\mathsf{add}(r_1, r_2, r, c)$, where c is an integer representing the carry, and which should hold when $[\![r_1]\!] + [\![r_2]\!] = [\![r]\!] + c$.

$$(\textsc{add})\ \frac{\mathsf{add}(r_1, r_2, r, c)}{\mathsf{add}(d_1{:}r_1, d_2{:}r_2, (x \mod b){:}r, x \div b)}\quad x = d_1 + d_2 + c$$

$$(\textsc{co-add})\ \frac{}{\mathsf{add}(r_1, r_2, r, c)}\quad c \in -1..2$$

In Section 3.5.1 it is proved that such definition is correct, that is, it correctly defines the addition between real numbers. It is also well-known that rational streams of digits represent rational numbers, that is, if r is a rational stream of digits, then $[\![r]\!]$ is a rational number. We show here that the regular interpretation of the above inference system with corules is correct with respect to the addition of rational numbers.

Define the set $\mathcal{S}^{\mathsf{add}}$ of correct judgements as follows: $\mathsf{add}(r_1, r_2, r, c) \in \mathcal{S}^{\mathsf{add}}$ iff r_1, r_2 and r are rational and $[\![r_1]\!] + [\![r_2]\!] = [\![r]\!] + c$. We start by proving completeness, stated below:

> for rational streams r_1, r_2, r, if $[\![r_1]\!] + [\![r_2]\!] = [\![r]\!] + c$,
> then $\langle \mathcal{I}^{\mathsf{add}}, \mathcal{I}^{\mathsf{add}}_{\mathsf{co}} \rangle \vdash_\rho \mathsf{add}(r_1, r_2, r, c)$.

We use the bounded regular coinduction principle. First of all, note that, since $[\![r]\!] \in [0, 1]$ for any stream r, if $\mathsf{add}(r_1, r_2, r, c) \in \mathcal{S}^{\mathsf{add}}$, then $c = [\![r_1]\!] + [\![r_2]\!] - [\![r]\!]$, hence $c \geq -1$ and $c \leq 2$. Therefore, we immediately have that all judgements in $\mathcal{S}^{\mathsf{add}}$ have a finite proof tree using also the coaxiom.

Assume $\mathsf{add}(r_1, r_2, r, c) \in \mathcal{S}^{\mathsf{add}}$ and define X as follows: $\mathsf{add}(s_1, s_2, s, c') \in X$ iff $[\![s_1]\!] + [\![s_2]\!] = [\![s]\!] + c'$ and s_1 and s_2 are substreams of r_1 and r_2, respectively. Trivially, $\mathsf{add}(r_1, r_2, r, c) \in X$ and X is finite because, since r_1 and r_2 are rational, they have finitely many different substreams, and c' can assume only four values, hence $[\![s]\!] = [\![s_1]\!] + [\![s_2]\!] - c'$ can assume only finitely many values, and so there are finitely many s satisfying that equation. Now we have to check that X is consistent. Assume $\mathsf{add}(d_1{:}s_1, d_2{:}s_2, d{:}s, c') \in X$, then we have $[\![d_1{:}s_1]\!] + [\![d_2{:}s_2]\!] = [\![d{:}s]\!] + c'$. It is easy to check that, for any stream t and digit d, $b[\![d{:}t]\!] = d + [\![t]\!]$. Hence, we get $[\![s_1]\!] + [\![s_2]\!] = [\![s]\!] + c''$, with $c'' = bc' + d - d_1 - d_2$. Since s_1 and s_2 are still substreams of r_1 and r_2, respectively, we get $\mathsf{add}(s_1, s_2, s, c'') \in X$, as needed.

We now prove soundness, as stated below:

> if $\langle \mathcal{I}^{\mathsf{add}}, \mathcal{I}^{\mathsf{add}}_{\mathsf{co}} \rangle \vdash_\rho \mathsf{add}(r_1, r_2, r, c)$,
> then $[\![r_1]\!] + [\![r_2]\!] = [\![r]\!] + c$ and r_1,r_2 and r are rational streams.

For any finite set H, we define $\mathcal{S}^{\mathsf{add}}_H$ as follows: $\mathsf{add}(r_1, r_2, r, c_0) \in \mathcal{S}^{\mathsf{add}}_H$ iff $r_1 = d_{11}{:}\ldots{:}d_{1n}{:}s_1$, $r_2 = d_{21}{:}\ldots{:}d_{2n}{:}s_2$, $r = d_1{:}\ldots{:}d_n{:}s$ and there are $c_1, \ldots, c_n \in -1..2$ such that, for all $i \in 1..n$, $d_{1i}+d_{2i}+c_i = bc_{i-1}+d_i$, and either $\mathsf{add}(s_1, s_2, s, c_n) \in H$ or $s_1 = r_1$, $s_2 = r_2$, $s = r$ and $c_0 = c_n$. The two closure properties in Proposition 8.37 are easy to check. Hence, to conclude it is enough we show that $\mathcal{S}^{\mathsf{add}}_\emptyset \subseteq \mathcal{S}^{\mathsf{add}}$. To this end, assume $\mathsf{add}(r_1, r_2, r, c_0) \in \mathcal{S}^{\mathsf{add}}_\emptyset$, then, by definition, we have $r_1 = d_{11}{:}\ldots{:}d_{1n}{:}r_1$, $r_2 = d_{21}{:}\ldots{:}d_{2n}{:}r_2$, $r = d_1{:}\ldots{:}d_n{:}r$ and there are

$c_1, \ldots, c_n \in -1..2$ such that, for all $i \in 1..n$, $d_{1i} + d_{2i} + c_i = bc_{i-1} + d_i$. This implies that $r_1 = (d_{1_i})_{i \in \mathbb{N}_{>0}}$, $r_2 = (d_{2i})_{i \in \mathbb{N}_{>0}}$ and $r = (d_i)_{i \in \mathbb{N}_{>0}}$ are rational streams and, for all $i \in \mathbb{N}_{>0}$, $d_{1i} + d_{2i} + c_{j+1} = bc_j + d_i$ where $j = i \mod n$. Hence, we have only to check that $[\![r_1]\!] + [\![r_2]\!] = [\![r]\!] + c_0$. We define sequences $(x_k)_{k \in \mathbb{N}_{>0}}$ and $(y_k)_{k \in \mathbb{N}_{>0}}$ as $x_k = \sum_{i=1}^{k} (d_{1i} + d_{2i}) b^{-i}$ and $y_k = \sum_{i=1}^{k} d_i b^{-i}$. Then we have to show that $\lim x_k - \lim y_k = \lim(x_k - y_k) = c_0$, because $[\![r_1]\!] + [\![r_2]\!] = \lim x_k$ and $[\![r]\!] = \lim y_k$. As, for all $k \in \mathbb{N}_{>0}$, we have $d_{1k} + d_{2k} + c_{j+1} = bc_j + d_k$ with $j = k \mod n$, we get $c_0 - (x_k - y_k) = c_0 + y_k - x_k = c_{j+1} b^{-k}$, that, when k tends to ∞, converges to 0, hence we get $\lim(x_k - y_k) = c_0$.

9

Flexible coinductive logic programming

Logic programming is a declarative programming paradigm based on Horn clause logic. Programs are sets of clauses, defining how to derive other atoms from given ones, and can have both an inductive and a coinductive semantics. Indeed, as we will formally recall in Section 9.1, clauses of a logic program can be seen as meta-rules of an inference system where judgments are ground atoms: inference rules are ground instances of clauses, and a ground atom is derivable if it has a finite proof tree in the inductive interpretation and a possibly infinite proof tree in the coinductive one.

As happens for inference systems (cf Chapter 3), standard inductive and coinductive semantics of logic programs sometimes are not enough to properly define predicates on possibly infinite terms (Simon et al., 2007; Ancona, 2013). Consider the logic program in Figure 9.1, defining some predicates on lists of numbers represented with the standard Prolog syntax. For simplicity, we consider built-in numbers, as in Prolog.

In standard logic programming, terms are inductively defined, that is, are finite, and predicates are inductively defined as well. In the example program, only finite lists are considered, such as, *e.g.*, `[1|[2|[]]]`, and the three predicates are correctly defined on such lists.

Coinductive logic programming (coLP), introduced by Simon (2006), extends standard logic programming with the ability of reasoning about infinite objects and their properties. Terms are coinductively defined, that is, can be infinite, and predicates are coinductively defined as well. In the example, also infinite lists, such as `[1|[2|[3|[4|...]]]]`, are considered, and the coinductive inter-

```
all_pos([])    ←
all_pos([N|L]) ← N>0, all_pos(L)
member(X,[X|_]) ←
member(X,[Y|L]) ← X≠Y, member(X,L)
maxElem([N],N)   ←
maxElem([N|L],M) ← maxElem(L,M1), M is max(N,M1)
```

FIGURE 9.1 An example of logic program: *all_pos*(l) succeeds iff l contains only positive numbers, *member*(x, l) iff x is in l, *maxElem*(l, x) iff x is the greatest number in l.

pretation of `all_pos` gives the expected meaning on such lists. However, this is not the case for the other two predicates: for `member` the correct interpretation is still the inductive one, as in the coinductive semantics `member(`x`,`l`)` always succeeds for an infinite list l. For instance, for L the infinite list of 0's, `member(1,`L`)` has an infinite proof tree where for each node we apply the second clause. Therefore, these two predicates cannot coexist in the same program, as they require two different interpretations.[1]

The predicate `maxElem` shows an even worse situation. As discussed in Chapter 3, the inductive semantics again does not work on infinite lists, but also the coinductive one is not correct: `maxElem(`l`,`n`)` succeeds whenever n is greater than all the elements of l. The expected meaning lies between the inductive and the coinductive semantics, hence, to get it, we need something beyond standard semantics.

The generalisation of inference systems by corules, presented in Chapter 3, is able to express a variety of intermediate interpretations. Viewing logic programs as particular inference systems and guided by this abstract setting, which provides solid foundations, we develop an extension of logic programming supporting flexible coinduction,

Syntactically, programs are enriched by *coclauses*, which resemble clauses but have a special meaning used to tune the interpretation of predicates. By adding coclauses, we can obtain a declarative semantics intermediate between the inductive and the coinductive one. Standard (inductive) and coinductive logic programming are subsumed by a particular choice of coclauses. Correspondingly, operational semantics is a combination of standard SLD resolution (Lloyd, 1987; Apt, 1997) and coSLD resolution as introduced by Simon (2006) and Simon et al. (2006, 2007). More precisely, as in coSLD resolution, it keeps trace of already considered goals, called *coinductive hypotheses*. However, when a goal unifying with a coinductive hypothesis is found, rather than being considered successful as in coSLD resolution, its standard SLD resolution is triggered in the program where also coclauses are considered. Our main result is that such operational semantics is *sound and complete* with respect to the declarative one restricted to the regular case, relying on results in Chapter 8.

An important additional result is that the operational semantics is not incidental, but, as the declarative semantics, turns out to correspond to a precise notion on the inference system denoted by the logic program. Indeed, as detailed in Chapter 8, given an inference system, we can always construct another one, with judgments enriched by circular hypotheses, which, interpreted inductively, is equivalent to the *regular* interpretation of the original inference system. In other words, there is a canonical way to derive a (semi-)algorithm to show that a judgment has a regular proof tree, and our operational semantics corresponds to this algorithm. This more abstract view supports the reliab-

1 To overcome this issue, Simon et al. (2007) introduce *co-logic programming*, allowing the programmer to mark predicates as either inductive or coinductive. The declarative semantics, however, becomes quite complex, because stratification is needed.

ility of the approach, and, indeed, the proof of equivalence with declarative semantics can be nicely done in a modular way, that is, by relying on a general result proved in Section 8.6.3.

Finally, we also have a prototype SWI-Prolog implementation[2] of the operational semantics designed in this chapter, which can be used to experiment with flexible coinductive logic programming.

The rest of the chapter is organised as follows. After basic notions in Section 9.1, in Section 9.2 we introduce logic programs with coclauses and their declarative semantics, and in Section 9.3 the operational semantics. We provide significant examples in Section 9.4 and the results in Section 9.5.

9.1 Logic programs as inference systems

We present (standard inductive and coinductive) logic programming (Lloyd, 1987; Apt, 1997; Simon, 2006; Simon et al., 2006, 2007) as a particular instance of the general semantic framework of inference systems (cf. Chapter 2).

Assume a *first order signature* $\langle \mathcal{P}, \mathcal{F}, \mathcal{V} \rangle$ with $\mathcal{P}$ set of *predicate symbols* p, $\mathcal{F}$ set of *function symbols* f, and $\mathcal{V}$ countably infinite set of *variable symbols* X (*variables* for short). Each symbol comes with its *arity*, a natural number denoting the number of arguments. Variables have arity 0. A function symbol with arity 0 is a *constant*.

Terms t, s, r are (possibly infinite) trees with nodes labeled by function or variable symbols, where the number of children of a node is the symbol arity[3]. *Atoms* A, B, C are (possibly infinite) trees with the root labeled by a predicate symbol and other nodes by function or variable symbols, again accordingly with the arity. Terms and atoms are *ground* if they do not contain variables, and *finite* (or *syntactic*) if they are finite trees. *(Definite) clauses* have shape $A \leftarrow B_1, \ldots, B_n$ with $n \geq 0$, A, B_1, …, B_n finite atoms. A clause where $n = 0$ is called a *fact*. A *(definite) logic program* P is a finite set of clauses.

Substitutions θ, σ are partial maps from variables to terms with a finite domain. We write $t\theta$ for the application of θ to a term t, call $t\theta$ an *instance* of t, and analogously for atoms, set of atoms, and clauses. A substitution θ is *ground* if, for all $X \in \mathrm{dom}(\theta)$, $\theta(X)$ is a ground term, *syntactic* if, for all $X \in \mathrm{dom}(\theta)$, $\theta(X)$ is a finite (syntactic) term.

In order to see a logic program P as an inference system, we fix as universe the *complete Herbrand base* HB_∞, that is, the set of all (finite and infinite) ground atoms[4]. Then, P can be seen as a set of *meta-rules* defining an inference system $\|P\|$ on HB_∞. That is, $\|P\|$ is the set of ground instances of clauses in P, where $A \leftarrow B_1, \ldots, B_n$ is seen as an inference rule $\langle \{B_1, \ldots, B_n\}, A \rangle$. In this way, typical notions related to declarative semantics of logic programs turn out

2 Available at https://github.com/davideancona/coLP-with-coclauses.

3 For a more formal definition based on paths see, e.g., the work of Ancona and Dovier (2015).

4 Traditionally (Lloyd, 1987), the inductive declarative semantics is restricted to finite atoms. We define also the inductive semantics on the complete Herbrand base in order to work in a uniform context.

to be instances of analogous notions for inference systems. Notably, the (one step) inference operator associated with a program $T_P : \wp(\mathsf{HB}_\infty) \to \wp(\mathsf{HB}_\infty)$, defined by:

$$T_P(I) = \{A \in \mathsf{HB}_\infty \mid (A \leftarrow B_1, \ldots, B_n) \in \|P\|, \{B_1, \ldots, B_n\} \subseteq I\}$$

is exactly $F_{\|P\|}$ (cf. Section 2.3). An *interpretation* of the signature, that is, a set $I \subseteq \mathsf{HB}_\infty$, is a *model* of a program P if $T_P(I) \subseteq I$, that is, it is closed with respect to $\|P\|$, and, dually, it is a *comodel* of a program P if $I \subseteq T_P(I)$, that is, it is consistent with respect to $\|P\|$ (cf. Definition 2.2). Then, the inductive declarative semantics of P is the least model of P and the coinductive declarative semantics[5] is the greatest comodel of P. These two semantics coincide with the inductive and coinductive interpretations of $\|P\|$ (cf. Definition 2.6 and Theorems 2.23 and 2.24), hence we denote them by $\mu[\![P]\!]$ and $\nu[\![P]\!]$, respectively.

9.2 Coclauses

We introduce logic programs with coclauses and define their declarative semantics. Consider again the example in Figure 9.1 where, as discussed in the introduction of the chapter, each predicate needed a different kind of interpretation.

As shown in the previous section, the logic program in Figure 9.1 can be seen as an inference system. In this context, flexible coinduction, introduced in Chapter 3, provides a generalisation able to overcome these limitations. The key notion are corules, special inference rules used to control the semantics of an inference system. Recall from Chapter 3 (cf. Definitions 3.1 and 3.3 and Theorem 3.13) that, given an inference system with corules $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$, its interpretation $\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$, is constructed in two steps.

- first, we take the inductive interpretation of the union $\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}$, that is, the smallest $(\mathcal{I} \cup \mathcal{I}_{\mathsf{co}})$-closed set, denoted by $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$,
- then, the union of all $\mathcal{I}$-consistent sets which are subsets of $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$, that is, the largest $\mathcal{I}$-consistent subset of $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$.

In proof-theoretic terms, $\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$ is the set of judgements with an arbitrary (finite or not) proof tree in $\mathcal{I}$, whose nodes all have a finite proof tree in $\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}$. Essentially, by corules we filter out some, undesired, infinite proof trees. Theorem 3.13 shows that $\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$ is a fixed point of $F_{\mathcal{I}}$.

To introduce flexible coinduction in logic programming, following this general framework, first we slightly extend the syntax by introducing *(definite) coclauses*, written $A \Leftarrow B_1, \ldots, B_n$, where A, B_1, …, B_n are finite atoms. A coclause where $n = 0$ is called a *cofact*. Coclauses syntactically resemble clauses, but are used in a special way, like corules for inference systems. More precisely, we have the following definition:

5 Introduced by Simon (2006) and Simon et al. (2006) to properly deal with predicates on infinite terms.

```
allPos([])     ←
allPpos([N|L]) ← N>0, all_pos(L)
allPos(_)     ⇐
member(X,[X|_]) ←
member(X,[Y|L]) ← X≠Y, member(X,L)
maxElem([N],N)   ←
maxElem([N|L],M) ← maxElem(L,M1), M is max(N,M1)
maxElem([N|L],N) ⇐
```

FIGURE 9.2 The logic program in Figure 9.1 enriched with coclauses.

DEFINITION 9.1 : A *logic program with coclauses* is a pair $\langle P, P_{\text{co}} \rangle$ where P and P_{co} are sets of clauses. Its *declarative semantics*, denoted by $\nu[\![P, P_{\text{co}}]\!]$, is the largest comodel of P which is a subset of $\mu[\![P \cup P_{\text{co}}]\!]$.

In other words, the declarative semantics of $\langle P, P_{\text{co}} \rangle$ is the coinductive semantics of P where, however, clauses are instantiated only on elements of $\mu[\![P \cup P_{\text{co}}]\!]$. Note that this is the interpretation of the generalised inference system $\langle \|P\|, \|P_{\text{co}}\| \rangle$.

In Figure 9.2, we report the version of the example in Figure 9.1, equipped with coclauses. In this way, all the predicate definitions are correct with respect to the expected semantics:

- `allPos` has coinductive semantics, as the coclause allows any infinite proof trees.
- `member` has inductive semantics, as without coclauses no infinite proof tree is allowed.
- `maxElem` has an intermediate semantics, as the coclause allows only infinite proof trees where nodes have shape `maxElem(`l`,`x`)` with x an element of l.

As the example shows, coclauses allow the programmer to mix inductive and coinductive predicates, and to correctly define predicates which are neither inductive, nor purely coinductive. For this reason we call this paradigm *flexible coinductive logic programming*. Note that, as shown for inference systems with corules (cf. Corollary 3.15), inductive and coinductive semantics are particular cases. Indeed, they can be recovered by special choices of coclauses: the former is obtained when no coclause is specified, while the latter when each atom in HB_∞ is an instance of the head of a cofact.

9.3 Big-step operational semantics

In this section we define an operational counterpart of the declarative semantics of logic programs with coclauses introduced in the previous section.

As in standard coLP (Simon, 2006; Simon et al., 2006, 2007), we use finite sets of equations between finite (syntactic) terms to represent possibly infinite

terms. For instance, the equation `L =:= [1,2|L]` represents the infinite list `[1,2,1,2,...]`.

Since the declarative semantics of logic programs with coclauses is a combination of inductive and coinductive semantics, their operational semantics combines standard SLD resolution (Lloyd, 1987; Apt, 1997) and coSLD resolution (Simon, 2006; Simon et al., 2006, 2007). It is presented, rather than in the traditional small-step style, in big-step style, as introduced by Ancona and Dovier (2015). This style turns out to be simpler since coinductive hypotheses (see below) can be kept local. Moreover, it naturally leads to an interpreter, and makes it simpler to prove its correctness with respect to declarative semantics (see the next section).

We introduce some notations. First of all, in this section we assume atoms and terms to be finite (syntactic). A *goal* is a pair $\langle G; E\rangle$, where G is a finite sequence of atoms. A goal is *empty* if G is the empty sequence, denoted ε, as usual. An *equation* has shape $s =:= t$ where s and t are terms, and we denote by E a finite set of equations.

Intuitively, a goal can be seen as a query to the program and the operational semantics has to compute answers (a.k.a. solutions) to such a query. More in detail, the operational semantics, given a goal $\langle G; E_1\rangle$, returns another set of equations E_2, which represents the answers to the goal. For instance, given the program in Figure 9.2, for the goal $\langle$`maxElem(L,M)`$;\{$`L =:= [1,2|L]`$\}\rangle$ the operational semantics returns the set of equations $\{$`L =:= [1,2|L]`$,$ `M =:= 2`$\}$.

The judgment of the operational semantics has shape

$$\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle G; E_1\rangle \Rightarrow E_2$$

meaning that resolution of $\langle G; E_1\rangle$, under the *coinductive hypotheses* S (Simon et al., 2006), succeeds in $\langle P, P_{\mathsf{co}}\rangle$, producing a set of equations E_2. Set $\mathsf{Var}(t)$ the set of variables in a term, and analogously for atoms, set of atoms, and equations. Resolution starts with no coinductive hypotheses, that is, the top-level judgment has shape $\langle P, P_{\mathsf{co}}\rangle; \emptyset \Vdash \langle G; E_1\rangle \Rightarrow E_2$. We assume $\mathsf{Var}(S) \subseteq \mathsf{Var}(E_1)$, modelling the intuition that S keeps track of already considered atoms. This condition holds for the initial judgement, and is preserved by rules in Figure 9.3, hence it is not restrictive.

The operational semantics has two flavours:

- If there are no coclauses ($P_{\mathsf{co}} = \emptyset$), then the judgment models standard SLD resolution, hence the set of coinductive hypotheses is not significant.
- Otherwise, the judgment models *flexible coSLD resolution*, which follows the same schema of coSLD resolution, in the sense that it keeps track in S of the already considered atoms. However, when an atom A in the current goal unifies with a coinductive hypothesis, rather than just considering A successful as in coSLD resolution, standard SLD resolution of A is triggered in the program $P \cup P_{\mathsf{co}}$, that is, also coclauses can be used.

The judgement is inductively defined by the rules in Figure 9.3, which rely on some auxiliary (standard) notions. A *solution* of an equation $s =:= t$ is a *unifier* of t and s, that is, a substitution θ such that $s\theta = t\theta$. A solution of a finite set of

$$\text{(EMPTY)}\ \frac{}{\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle \varepsilon; E\rangle \Rightarrow E}$$

$$\text{(CO-HYP)}\ \frac{\begin{array}{l}\langle P \cup P_{\mathsf{co}}, \emptyset\rangle; \emptyset \Vdash \langle A; E_1 \cup E_{A,B}\rangle \Rightarrow E_2 \\ \langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle G_1, G_2; E_2\rangle \Rightarrow E_3\end{array}}{\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle G_1, A, G_2; E_1\rangle \Rightarrow E_3}\quad \begin{array}{l} B \in S \\ E_1 \vdash A = B \\ P_{\mathsf{co}} \neq \emptyset\end{array}$$

$$\text{(STEP)}\ \frac{\begin{array}{l}\langle P, P_{\mathsf{co}}\rangle; S \cup \{A\} \Vdash \langle C_1, \ldots, C_n; E_1 \cup E_{A,B}\rangle \Rightarrow E_2 \\ \langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle G_1, G_2; E_2\rangle \Rightarrow E_3\end{array}}{\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle G_1, A, G_2; E_1\rangle \Rightarrow E_3}\quad \begin{array}{l} \theta \text{ fresh renaming} \\ B\theta \leftarrow C_1\theta, \ldots, C_n\theta \in P \\ E_1 \vdash A = B\end{array}$$

FIGURE 9.3 Big-step operational semantics of flexible coinductive logic programming

equations E is a solution of all the equations in E and E is *solvable* if there exists a solution of E. Two atoms A and B are *unifiable* in a set of equations E, written $E \vdash A = B$, if $A = p(s_1, \ldots, s_n)$, $B = p(t_1, \ldots, t_n)$ and $E \cup \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ is solvable, and we denote by $E_{A,B}$ the set $\{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$.

Rule (EMPTY) states that the resolution of an empty goal succeeds. In rule (STEP), an atom A to be resolved is selected, and a clause of the program P is chosen such that A unifies with the head of the clause in the current set of equations. Then, resolution of the original goal succeeds if both the body of the selected clause and the remaining atoms are resolved, enriching the set of equations correspondingly. As customary, the selected clause is renamed using fresh variables, to avoid variable clashes in the set of equations obtained after unification. Note that, in the resolution of the body of the clause, the selected atom is added to the current set of coinductive hypotheses. This is not relevant for standard SLD resolution ($P_{\mathsf{co}} = \emptyset$). However, if $P_{\mathsf{co}} \neq \emptyset$, this allows rule (CO-HYP) to handle the case when an atom A that has to be resolved unifies with a coinductive hypothesis in the current set of equations. In this case, standard SLD resolution of such atom in the program $P \cup P_{\mathsf{co}}$ is triggered, and resolution of the original goal succeeds if both such standard SLD resolution of the selected atom and resolution of the remaining goal succeed.

In Figure 9.4 we show an example of resolution in the program of Figure 9.2. We use the shorter syntax `M=max(N1,N2)` instead of `M is max(N1,N2)`, abbreviate by `mE` the predicate `maxElem` and by (S), (E), (C) the rules (STEP), (EMPTY) and (CO-HYP), respectively. When applying rule (STEP), we also indicate the clause/coclause which has been used: we write 1,2,3 for the two clauses and the coclause for the `maxElem` predicate (the first clause is never used in this example). Finally, to keep the example readable and focus on key aspects, we make some simplifications: notably, T_m stands for an omitted proof tree solving atoms of shape `_ is max(_,_)`, irrelevant applications of rule (EMPTY) are omitted and, morever, equations between lists are implicitly applied.

As final remark, note that flexible coSLD resolution nicely subsumes both SLD and coSLD. The former, as already said, is obtained when the set of

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle\{1,2\},3\rangle;\texttt{mE(L,M)},\ldots \Vdash \langle \texttt{mE(L,M2)},\texttt{M1=max(2,M2)};eq_{\texttt{L}}\rangle \Rightarrow eqs_3}\,(\texttt{C})\;T_c \quad T_m}{\langle\{1,2\},3\rangle;\texttt{mE(L,M)} \Vdash \langle \texttt{mE([2|L],M1)},\texttt{M=max(1,M1)};eq_{\texttt{L}}\rangle \Rightarrow eqs_3,\texttt{M=2}}\,(\texttt{S-2})\;T_m}{\langle\{1,2\},3\rangle;\emptyset \Vdash \langle \texttt{mE(L,M)};eq_{\texttt{L}}\rangle \Rightarrow eqs_3,,\texttt{M=2}}\,(\texttt{S-2})\;T_m$$

$$T_c = (\texttt{S-2})\;\dfrac{\dfrac{\dfrac{}{\langle\{1,2,3\},\emptyset\rangle;\ldots \Vdash \langle \varepsilon;eqs_1,\texttt{M3=2}\rangle \Rightarrow eqs_1,\texttt{M3=2}}\,(\texttt{E})\;T_m}{\langle\{1,2,3\},\emptyset\rangle;\ldots \Vdash \langle \texttt{mE([2|L],M3)},\texttt{M2=max(1,M3)};eqs_1\rangle \Rightarrow eqs_2}\,(\texttt{S-3})}{\langle\{1,2,3\},\emptyset\rangle;\emptyset \Vdash \langle \texttt{mE(L,M2)};eqs_1\rangle \Rightarrow eqs_2}$$

$$eq_{\texttt{L}} = \texttt{L=[1,2|L]} \qquad eqs_1 = eq_{\texttt{L}},\texttt{M2=M}$$
$$eqs_2 = eqs_1\texttt{M3=2},\texttt{M2=2} \qquad eqs_3 = eqs_2,\texttt{M1=2}$$

FIGURE 9.4 Example of resolution of `maxElem(L,M)` with `L = [1,2|L]`.

coclauses is empty, that is, the program is inductive. The latter is obtained when, for all predicate p of arity n, we have a cofact $p(X_1, \ldots, X_n) \Leftarrow$.

9.4 Examples

In this section we discuss some more sophisticated examples.[6]

∞-REGULAR EXPRESSIONS We define *∞-regular expressions* on an alphabet Σ, a variant of the formalism defined by Löding and Tollkötter (2016) for denoting languages of finite and infinite words, the latter also called ω-words, as follows:

$$r ::= \emptyset \mid \epsilon \mid a \mid r_1 \cdot r_2 \mid r_1 + r_2 \mid r^\star \mid r^\omega$$

where $a \in \Sigma$. The syntax of standard regular expressions is extended by r^ω, denoting the ω-power of the language A_r denoted by r. That is, the set of words obtained by concatenating infinitely many times words in A_r. In this way, we can denote also languages containing infinite words.

In Figure 9.5 we define the predicate `match`, such that `match(w,r)` holds if the finite or infinite word w, implemented as a list, belongs to the language denoted by r. For simplicity, we consider words over the alphabet $\{0, 1\}$.

Concatenation of words needs to be defined coinductively, to correctly work on infinite words as well. Note that, when w_1 is infinite, $w_1 w_2$ is equal to w_1.

On operators of regular expressions, `match` can be defined in the standard way (no coclauses). In particular, the definition for expressions of shape $r^\star$ follows the explicit definition of the $\star$-closure of a language: given a language L, a word w belongs to $L^\star$ iff it can be decomposed as $w_1 \ldots w_n$, for some $n \geq 0$, where $n = 0$ means w is empty, and $w_i \in L$, for all $i \in 1..n$. This condition is checked by the auxiliary predicate `match_star`.

To define when a word w matches r^ω we have two cases. If w is empty, then it is enough to check that the empty word matches r, as expressed by the first clause, because concatenating infinitely many times the empty word

6 Executable code of the examples is available at https://github.com/davideancona/coLP-with-coclauses.

```
concat([],W,W) ←
concat([B|W1],W2,[B|W3]) ← concat(W1,W2,W3)
concat(W1,W2,W1) ⇐
match([],eps) ←
match([0],0) ←
match([1],1) ←
match(W,cat(R1,R2)) ← match(W1,R1), match(W2,R2), concat(W1,W2,W
    )
match(W,plus(R1,R2)) ← match(W,R1)
match(W,plus(R1,R2)) ← match(W,R2)
match(W,star(R)) ← match_star(N,W,R)
match([],omega(R)) ← match([],R)
match([B|W],omega(R)) ← match([B|W1],R), match(W2,omega(R)),
    concat(W1,W2,W)
match(W,omega(R)) ⇐
match_star(0,[],R)   ←
match_star(s(N),W,R) ← match(W1,R), match_star(N,W2,R), concat(
    W1,W2,W)
```

FIGURE 9.5 A logic program for ∞-regular expression recognition.

we get again the empty word. Otherwise, we have to decompose w as $w_1 w_2$ where w_1 is not empty and matches r and w_2 matches r^ω as well, as formally expressed by the second clause. To propertly handle infinite words, we need to concatenate infinitely many non-empty words, hence we need to apply the second clause infinitely many times. The coclause allows all such infinite derivations.

AN LTL FRAGMENT In Figure 9.6 we define the predicate `sat` such that `sat(`w`,`φ`)` succeeds iff the ω-word w over the alphabet $\{0, 1\}$ satisfies the formula φ of the fragment of the Linear Temporal Logic with the temporal operators `until` (**U**) and `always` (**G**) and the predicate `zero` and its negation[7] `one`.

Since `sat(`w`,always(`φ`))` holds iff all *infinite* suffixes of ω-words w satisfy formula φ, infinite derivations has to be considered, hence a coclause is needed. For instance, `sat(W0,always(zero))`, with `W0=[0|W0]`, succeeds because the atom `sat(W0,always(zero))` in the body of the clause for `always` unifies[8] with the coinductive hypothesis `sat(W0,always(zero))` (see rule (CO-HYP) in Figure 9.3) and the coclause allows it to succeed with respect to standard SLD resolution. Further, the atom `sat(W0,zero)` in the body succeeds, thanks to the first fact in the logic program.

Differently to `always`, the satisfaction of `until` has not to use infinite derivations, because `until(`φ_1`,`φ_2`)` holds iff φ_2 is satisfied after a *finite* number of steps; for this reason, no coclause is given for this operator. For instance, `sat([1,1,0|W1],until(one,zero))` with `W1=[1|W1]` succeeds with respect

7 Predicates `true` and `false` could be easily defined as well.

8 Actually, in this case the atom to be resolved and the coinductive hypothesis are syntactically equal.

```
sat_exists(0,W,Ph) ← sat(W,Ph)
sat_exists(s(N),[B|W],Ph) ← sat_exists(N,W,Ph)
sat_all(0,W,Ph) ←
sat_all(s(N),[B|W],Ph) ← sat([B|W],Ph), sat_all(N,W,Ph)
sat([0|W],zero) ←
sat([1|W],one)  ←
sat([B|W],always(Ph)) ← sat([B|W],Ph), sat(W,always(Ph))
sat(W,always(Ph)) ⇐
sat([B|W],until(Ph1,Ph2)) ← sat_exists(N,[B|W],Ph2), sat_all(N,[
    B|W],Ph1)
```

FIGURE 9.6 A logic program for satisfaction of an LTL fragment: `sat_exists(N,W,Ph)` succeeds iff suffix at `N` of ω-word `W` satisfies `Ph`, `sat_all(N,W,Ph)` succeeds iff all suffixes of ω-word `W` at index `< N` satisfy `Ph`, `sat(W,Ph)` succeeds iff ω-word `W` satisfies `Ph`.

to standard SLD resolution, while `sat(W1,until(one,zero))`, `sat(W1,until(always(one),zero))`, and `sat(W1,until(always(one),always(zero)))` fail. The clause for `sat([B|W],until(Ph_1,Ph_2))` follows the standard definition of satisfaction for the **U** operator: there must exist a suffix of `[B|W]` at index `N` satisfying `Ph2` (`sat_exists(N,[B|W],Ph2)`) such that all suffixes of `[B|W]` at index less than `N` satisfy `Ph1` (`sat_all(N,[B|W],Ph1)`).

An interesting example concerns the goal `sat([1,1|W0],until(one,always(zero)))`, where the two temporal operators are mixed together: it succeeds as expected, thanks to the two clauses for `until` and the fact that `sat(W0,always(zero))` succeeds, as shown above.

Some of the issues faced in this example are also discussed by Gupta et al. (2011).

BIG-STEP SEMANTICS MODELING INFINITE BEHAVIOUR AND OBSERVATIONS Defining a big-step operational semantics modelling divergence is a difficult task, especially in presence of observations. In Chapter 6 we have shown how corules can be successfully employed to tackle this problem, providing examples of big-step semantics able to model divergence for several variations of the λ-calculus and different kinds of observations. Following this approach, we present in Figure 9.7 a similar example, but simpler, to keep it shorter: a logic program with coclauses defining the big-step semantics of a toy language able to output possibly infinite sequences[9] of integers. Expressions are regular terms generated by the following grammar:

$$e ::= skip \mid out(n) \mid seq(e_1, e_2)$$

where *skip* is the idle expression, $out(n)$ outputs n, and $seq(e_1, e_2)$ is the sequential composition. The semantic judgement has shape $e \Rightarrow \langle r, s \rangle$, represented by the atom `eval(e,r,s)`, where e is an expression, r is either *end* or *div*, for

9 For simplicity we consider only integers, but in fact the definition below allows any term as output.

```
concat([],S,S) ←
concat([N|S1],S2,[N|S3]) ← concat(S1,S2,S3)
eval(skip,end,[]) ←
eval(out(N),end,[N]) ←
eval(seq(E1,E2),R,S) ← eval(E1,end,S1), eval(E2,R,S2), concat(S1
    ,S2,S)
eval(seq(E1,E2),div,S) ← eval(E1,div,S)
eval(E,div,[]) ⇐
eval(seq(E1,E2),div,S) ⇐ eval(E1,end,[N|S1]), concat([N|S1],S2,S
    )
```

FIGURE 9.7 A logic program defining a big-step semantics with infinite behaviour and observations.

converging or diverging computations, respectively, and s is a possibly infinite sequence of integers.

Clauses for `concat` are pretty standard; in this case the definition is purely inductive (hence, no coclause is needed), since the left operand of concatenation is always a finite sequence. Clauses for `eval` are rather straightforward, but sequential composition $seq(e_1, e_2)$ deserves some comment: if the evaluation of e_1 converges, then the computation can continue with the evaluation of e_2, otherwise the overall computation diverges and e_2 is not evaluated.

As opposite to the previous examples, here we do not need just cofacts, but also a coclause; both the cofact and the coclause ensure that for infinite derivations only *div* can be derived. Furthermore, the cofact handles diverging expressions which produce a finite output sequence, as in `eval(E,div,[])` or in `eval(seq(out(1),E),div,[1])`, with `E=seq(skip,E)` or `E=seq(E,E)`, while the coclause deals with diverging expressions with infinite outputs, as in `eval(E,div,S)` with `E=seq(out(1),E)` and `S=[1|S]`. The body of the coclause ensures that the left operand of sequential composition converges, thus ensuring a correct productive definition.

9.5 Soundness and completeness

After formally relating the two approaches, we state soundness of the operational semantics with respect to the declarative one. Then, we show that completeness does not hold in general, and define the *regular* version of the declarative semantics. Finally, we show that the operational semantics is equivalent to this restricted declarative semantics.

RELATION BETWEEN OPERATIONAL AND DECLARATIVE SEMANTICS

As in the standard case, the first step is to bridge the gap between the two approaches: the former computing equations, the latter defining truth of atoms. This can be achieved through the notions of *answers* to a goal.

Given a set of equations E, $\mathsf{sol}(E)$ is the set of the *solutions* of E, that is, the

ground substitutions unifying all the equations in E. Then, $\theta \in \mathsf{sol}(E)$ is an *answer* to $\langle G; E\rangle$ if $\mathsf{Var}(G) \subseteq \mathsf{dom}(\theta)$.

The judgment $\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle G; E_1\rangle \Rrightarrow E_2$ described in Section 9.3 computes a set of answers to the input goal. Indeed, solutions of the output set of equations are solutions of the input set as well, since the following proposition holds.

PROPOSITION 9.2 : The following hold:

1. If $\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle G; E_1\rangle \Rrightarrow E_2$, then $E_1 \subseteq E_2$ and $\mathsf{Var}(G) \subseteq \mathsf{Var}(E_2)$.
2. If $E_1 \subseteq E_2$, then $\mathsf{sol}(E_2) \subseteq \mathsf{sol}(E_1)$.

Proof: Item 1 follows by a straightforward induction on rules in Figure 9.3, while Item 2 is trivial. □

On the other hand, we can define which answers are correct in an interpretation:

DEFINITION 9.3 : Let $I \subseteq \mathsf{HB}_\infty$ be an interpretation, the set of answers to $\langle G; E\rangle$ *correct in* I is

$$\mathsf{ans}(G, E, I) = \{\theta \in \mathsf{sol}(E) \mid G\theta \subseteq I\}$$

Hence, soundness of the operational semantics can be expressed as follows: all the answers computed for a given goal are correct in the declarative semantics.

THEOREM 9.4 : If $\langle P, P_{\mathsf{co}}\rangle; \emptyset \Vdash \langle G; E\rangle \Rrightarrow E'$ holds, then $\mathsf{sol}(E') \subseteq \mathsf{ans}(G, E, \nu[\![P, P_{\mathsf{co}}]\!])$.

COMPLETENESS ISSUES The converse of this theorem, that is, all correct answers can be computed, cannot hold in general, since, as shown by Ancona and Dovier (2015), coinductive declarative semantics does not admit any complete procedure[10], hence our model as well, since it generalizes the coinductive one. To explain why completeness does not hold in our case, we can adapt the following example from Ancona and Dovier (2015)[11], where p is a predicate symbol of arity 1, z and s are function symbols of arity 0 and 1 respectively.

```
p(X) ← p(s(X))
p(X) ⇐
```

Let us define $\underline{0} = z$, $\underline{n+1} = s(\underline{n})$ and $\underline{\omega} = s(s(\ldots))$. The declarative semantics is the set $\{p(\underline{x}) \mid x \in \mathbb{N} \cup \{\omega\}\}$. In the operational semantics, instead, only $p(\underline{\omega})$ is considered true. Indeed, all derivations have to apply the rule (CO-HYP), which imposes the equation $X \doteq s(X)$, whose unique solution is $\underline{\omega}$. Therefore, the operational semantics is not complete.

10 That is, establishing whether an atom belongs to the coinductive declarative semantics is neither decidable nor semi-decidable, even when the Herbrand universe is restricted to the set of rational terms.

11 Example 10 at page 8.

Now the question is the following: can we characterize in a declarative way answers computed by the big-step semantics? In the example, there is a difference between the atoms $p(\underline{\omega})$ and $p(\underline{n})$, with $n \in \mathbb{N}$, because the former has a regular proof tree, namely, a tree with finitely many different subtrees, while the latter has only non-regular, thus infinite, proof trees.

Following this observation, we prove that the operational semantics is sound and complete with respect to the restriction of the declarative semantics to atoms derivable by regular proof trees. As we will see, this set can be defined in model-theoretic terms, by restricting to finite comodels of the program, as done in Chapter 8 for an arbitrary (generalized) inference system. Here we rephrase definitions and results from Chapter 8 in the specific case of logic programs.

REGULAR DECLARATIVE SEMANTICS Let us write $X \subseteq_{\mathit{fin}} Y$ if X is a finite subset of Y. Recall that the regular interpretation of a generalised inference system $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$ (cf. Definitions 8.3 and 8.22 and Proposition 8.29) can be expressed as

$$\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] = \bigcup \{X \subseteq_{\mathit{fin}} \mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!] \mid X \subseteq F_{\mathcal{I}}(X)\}$$

This characterisation is like the one of $\nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$, except that we take the union only of those consistent subsets of $\mu[\![\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}]\!]$ which are *finite*. The set $\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$ is a fixed point of $F_{\mathcal{I}}$ (cf. Proposition 8.29) and, hence, we get $\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!] \subseteq \nu[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$.

The proof-theoretic characterization relies on regular proof trees, which are proof trees with a finite number of subtrees (Courcelle, 1983). That is, $\rho[\![\mathcal{I}, \mathcal{I}_{\mathsf{co}}]\!]$ is the set of judgments with a regular proof tree in $\mathcal{I}$ whose nodes all have a finite proof tree in $\mathcal{I} \cup \mathcal{I}_{\mathsf{co}}$.

As special case, we get regular semantics of logic programs with coclauses.

DEFINITION 9.5 : The *regular declarative semantics* of $\langle P, P_{\mathsf{co}} \rangle$, denoted by $\rho[\![P, P_{\mathsf{co}}]\!]$, is the union of all finite comodels of P which are a subset of $\mu[\![P \cup P_{\mathsf{co}}]\!]$.

As above, $\rho[\![P, P_{\mathsf{co}}]\!] \subseteq \nu[\![P, P_{\mathsf{co}}]\!]$, hence $\mathsf{ans}(G, E, \rho[\![P, P_{\mathsf{co}}]\!]) \subseteq \mathsf{ans}(G, E, \nu[\![P, P_{\mathsf{co}}]\!])$.

We state now soundness and completeness of the operational semantics with respect to this semantics. We write $\theta \preceq \sigma$ iff $\mathsf{dom}(\theta) \subseteq \mathsf{dom}(\sigma)$ and, for all $X \in \mathsf{dom}(\theta)$, $\theta(X) = \sigma(X)$. It is easy to see that $\preceq$ is a partial order and, if $\theta \preceq \sigma$ and $\mathsf{Var}(G) \subseteq \mathsf{dom}(\theta)$, then $G\theta = G\sigma$.

THEOREM 9.6 : If $\langle P, P_{\mathsf{co}} \rangle; \emptyset \Vdash \langle G; E \rangle \Rightarrow E'$, and $\theta \in \mathsf{sol}(E')$, then $\theta \in \mathsf{ans}(G, E, \rho[\![P, P_{\mathsf{co}}]\!])$.

THEOREM 9.7 : If $\theta \in \mathsf{ans}(G, E, \rho[\![P, P_{\mathsf{co}}]\!])$, then $\langle P, P_{\mathsf{co}} \rangle; \emptyset \Vdash \langle G; E \rangle \Rightarrow E'$, and $\theta \preceq \sigma$ for some E' and $\sigma \in \mathsf{sol}(E')$.

That is, any answer computed for a given goal is correct in the regular declarative semantics, and any correct answer in the regular declarative semantics is included in a computed answer. Theorem 9.6 immediately entails Theorem 9.4 as $\mathsf{ans}(G, E, \rho[\![P, P_{\mathsf{co}}]\!]) \subseteq \mathsf{ans}(G, E, \nu[\![P, P_{\mathsf{co}}]\!])$.

PROOF TECHNIQUE In order to prove the equivalence of the two semantics, we rely on a property which holds in general for the regular interpretation: we can construct an equivalent inductive characterization, see Section 8.6.3. That is, given a generalized inference system $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$ on the universe $\mathcal{U}$, we can construct an inference system $\mathcal{I}^{\cup \mathcal{I}_{\mathsf{co}}}$ with judgments of shape $H \triangleright j$, for $j \in \mathcal{U}$ and $H \subseteq_{fin} \mathcal{U}$, such that the inductive interpretation of $\mathcal{I}^{\cup \mathcal{I}_{\mathsf{co}}}$ coincides with the regular interpretation of $\langle \mathcal{I}, \mathcal{I}_{\mathsf{co}} \rangle$. The set H, whose elements are called *circular hypotheses*, is used to detect cycles in the proof.

In particular, for logic programs with coclauses, we get an inference system with judgments of shape $S \triangleright A$, for S finite set of ground atoms, and A ground atom, defined as follows. For reader's convenience, we report below the instantiation of Definition 8.31 and Corollary 8.34 for logic programs with coclauses. The circular hypotheses are called coinductive hypotheses to be uniform in this setting.

DEFINITION 9.8 : Given $\langle P, P_{\mathsf{co}} \rangle$, the inference system $P^{\cup P_{\mathsf{co}}}$ consists of the following (meta-)rules:

$$\text{(B-HP)}\ \frac{A \in S}{S \triangleright A}\quad A \in \mu[\![P \cup P_{\mathsf{co}}]\!]$$

$$\text{(B-UNFOLD)}\ \frac{S \cup \{A\} \triangleright B_1 \ \ldots\ S \cup \{A\} \triangleright B_n}{S \triangleright A}\quad (A \leftarrow B_1, \ldots, B_n) \in \|P\|$$

COROLLARY 9.9 : $P^{\cup P_{\mathsf{co}}} \vdash_{\mu} \emptyset \triangleright A$ iff $A \in \rho[\![P, P_{\mathsf{co}}]\!]$.

Note that the definition of $S \triangleright A$ by rules in $P^{\cup P_{\mathsf{co}}}$ has many analogies with that of the operational semantics in Figure 9.3. The key difference is that the former handles *ground*, not necessarily finite, atoms, the latter not necessarily ground finite atoms (we use the same metavariables A and S for simplicity). In both cases already considered atoms are kept in an auxiliary set S. In the former, to derive an atom $A \in S$, the side condition requires A to belong to the inductive intepretation of the program $P \cup P_{\mathsf{co}}$. In the latter, when an atom A *unifies* with one in S, standard SLD resolution is triggered in the program $P \cup P_{\mathsf{co}}$.

To summarize, $P^{\cup P_{\mathsf{co}}}$ can be seen as an abstract version, at the level of the underlying inference system, of operational semantics. Hence, the proof of soundness and completeness can be based on proving a precise correspondence between these two inference systems, both interpreted inductively. This is very convenient since the proof can be driven in both directions by induction on the defining rules.

The correspondence is formally stated in the following two lemmas.

LEMMA 9.10 : For all S and $\langle A_1, \ldots, A_n; E \rangle$, if $\langle P, P_{\mathsf{co}} \rangle; S \Vdash \langle A_1, \ldots, A_n; E \rangle \Rightarrow E'$ then, for all $\theta \in \mathsf{sol}(E')$ and $i \in 1..n$, $P^{\cup P_{\mathsf{co}}} \vdash_{\mu} S\theta \triangleright A_i\theta$.

LEMMA 9.11 : For all S, $\langle A_1, \ldots, A_n; E \rangle$ and $\theta \in \mathsf{sol}(E)$, if $P^{\cup P_{\mathsf{co}}} \vdash_{\mu} S\theta \triangleright A_i\theta$, for all $i \in 1..n$, then $\langle P, P_{\mathsf{co}} \rangle; S \Vdash \langle A_1, \ldots, A_n; E \rangle \Rightarrow E'$ and $\theta \preceq \sigma$, for some E' and $\sigma \in \mathsf{sol}(E')$.

Soundness follows from Lemma 9.10 and Corollary 9.9, as detailed below.

Proof (Theorem 9.6)*:* Let us assume $\langle P, P_{\mathsf{co}}\rangle; \emptyset \Vdash \langle G; E\rangle \Rrightarrow E'$ with $G = A_1, \ldots, A_n$, and consider $\theta \in \mathsf{sol}(E')$. By Lemma 9.10, for all $i \in 1..n$, $P^{\circlearrowleft P_{\mathsf{co}}} \vdash_\mu \emptyset \triangleright A_i\theta$ holds, hence, by Corollary 9.9, we get $A_i\theta \in \rho[\![P, P_{\mathsf{co}}]\!]$. Therefore, by Definition 9.5, we get $\theta \in \mathsf{ans}(G, E, \rho[\![P, P_{\mathsf{co}}]\!])$, as needed. □

Analogously, completeness follows from Lemma 9.11 and Corollary 9.9, as detailed below.

Proof (Theorem 9.7)*:* Let $G = A_1, \ldots, A_n$ and $\theta \in \mathsf{ans}(G, E, \rho[\![P, P_{\mathsf{co}}]\!])$. Then, for all $i \in 1..n$, we have $A_i\theta \in \rho[\![P, P_{\mathsf{co}}]\!]$ and, by Corollary 9.9, we get $P^{\circlearrowleft P_{\mathsf{co}}} \vdash_\mu \emptyset \triangleright A_i\theta$. Hence, the thesis follows by Lemma 9.11. □

We now prove Lemmas 9.10 and 9.11 and auxiliary results.

Let us start by Lemma 9.10. To carry out the proof, we rely on Corollary 9.9 and on the following proposition, stating that the inductive declarative semantics of a logic program coincides with the regular semantics of a logic program with no coclauses, and follows immediately from an analogous result in the general setting of inference systems (cf. Proposition 8.30).

PROPOSITION 9.12 : Let P be a logic program, then $\mu[\![P]\!] = \rho[\![P, \emptyset]\!]$.

Proof (Lemma 9.10)*:* The proof is by induction on rules of Figure 9.3.

Case: (EMPTY) There is nothing to prove.

Case: (STEP) We have $G = G_1, A_i, G_2$, there is a fresh renaming $B \leftarrow B_1, \ldots, B_k$ of a clause in P such that A_i and B are unifiable in E, that is, $E_1 = E \cup E_{A_i,B}$ is solvable, and $\langle P, P_{\mathsf{co}}\rangle; S \cup \{A_i\} \Vdash \langle B_1, \ldots, B_k; E_1\rangle \Rrightarrow E_2$ and $\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle G_1, G_2; E_2\rangle \Rrightarrow E'$ hold. Let $\theta \in \mathsf{sol}(E')$, then, by induction hypothesis, we have, for all $j \in 1..n$ with $j \neq i$, $P^{\circlearrowleft P_{\mathsf{co}}} \vdash_\mu S\theta \triangleright A_j\theta$ holds. By Proposition 9.2, we have $E_1 \subseteq E_2 \subseteq E'$, hence $\mathsf{sol}(E') \subseteq \mathsf{sol}(E_2) \subseteq \mathsf{sol}(E_1)$, thus $\theta \in \mathsf{sol}(E_2) \subseteq \mathsf{sol}(E_1)$, and, since $E_{A_i,B} \subseteq E_1$, θ is a unifier of A_i and B, that is, $A_i\theta = B\theta$. Then, by induction hypothesis, we also get, for all $j \in 1..k$, $P^{\circlearrowleft P_{\mathsf{co}}} \vdash_\mu (S \cup \{A_i\})\theta \triangleright B_j\theta$ holds. Since $(S \cup \{A_i\})\theta = S\theta \cup \{A_i\theta\}$ and $B\theta \leftarrow B_1\theta, \ldots, B_k\theta \in \|P\|$ and $A_i\theta = B\theta$, by rule (UNFOLD) of Definition 9.8, we get that $P^{\circlearrowleft P_{\mathsf{co}}} \vdash_\mu S\theta \triangleright A_i\theta$ holds as well.

Case: (CO-HYP) We have $G = G_1, A_i, G_2$, there is an atom $B \in S$ that unifies with A_i in E, that is, $E_1 = E \cup E_{A_i,B}$ is solvable, and $\langle P \cup P_{\mathsf{co}}, \emptyset\rangle; \emptyset \Vdash \langle A_i; E_1\rangle \Rrightarrow E_2$ and $\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle G_1, G_2; E_2\rangle \Rrightarrow E'$ hold. Let $\theta \in \mathsf{sol}(E')$, then, by induction hypothesis, we get, for all $j \in 1..n$ with $j \neq i$, $P^{\circlearrowleft P_{\mathsf{co}}} \vdash_\mu S\theta \triangleright A_j\theta$ holds. By Proposition 9.2, we have $E_1 \subseteq E_2 \subseteq E'$, hence $\mathsf{sol}(E') \subseteq \mathsf{sol}(E_2) \subseteq \mathsf{sol}(E_1)$, thus $\theta \in \mathsf{sol}(E_2) \subseteq \mathsf{sol}(E_1)$, and, since $E_{A_i,B} \subseteq E_1$, θ is a unifier of A_i and B, that is, $A_i\theta = B\theta$. By induction hypothesis, we get $(P \cup P_{\mathsf{co}})^{\circlearrowleft \emptyset} \vdash_\mu \emptyset \triangleright A_i\theta$, hence, by Corollary 9.9 and Proposition 9.12, we get $A_i\theta \in \mu[\![P \cup P_{\mathsf{co}}]\!]$. Furthermore, since $A_i\theta = B\theta$ and $B \in S$, we have $A_i\theta \in S\theta$. Therefore, by rule (HP) of Definition 9.8, we get that

$P^{\cup P_{\mathsf{co}}} \vdash_\mu S\theta \triangleright A_i\theta$ holds as well.

□

We now focus on Lemma 9.11. We need some preliminary results.

We start by observing a property of the operational semantics. In the following, we say that substitutions θ_1 and θ_2 are compatible, denoted by $\theta_1 \| \theta_2$ if, for all $X \in \mathsf{dom}(\theta_1) \cap \mathsf{dom}(\theta_2)$, $\theta_1(X) = \theta_2(X)$, and we denote by $\theta_1 \uplus \theta_2$ the union of two substitutions, which is well-defined only for compatible substitutions. Note that, $\theta_i \preceq \theta_1 \uplus \theta_2$, for all $i = 1, 2$, by definition.

PROPOSITION 9.13 : Let $\langle G; E_1 \rangle$ be a goal, $\theta_1 \in \mathsf{sol}(E_1)$, E_1' such that $E_1 \subseteq E_1'$ and $\theta_1 \preceq \sigma_1$, for some $\sigma_1 \in \mathsf{sol}(E_1')$. If $\langle P, P_{\mathsf{co}} \rangle; S \Vdash \langle G; E_1 \rangle \Rrightarrow E_2$ and $\theta_1 \preceq \theta_2$, for some $\theta_2 \in \mathsf{sol}(E_2)$, then there exists E_2' such that $E_2 \subseteq E_2'$, $\langle P, P_{\mathsf{co}} \rangle; S \Vdash \langle G; E_1' \rangle \Rrightarrow E_2'$ and $\sigma_1 \preceq \sigma_2$, for some $\sigma_2 \in \mathsf{sol}(E_2')$.

Proof: The proof is by induction on the big-step rules in Figure 9.3.

Case: (EMPTY) We have $E_1 = E_2$, hence the thesis follows by taking $E_2' = E_1'$.

Case: (STEP) We know that $G = G_1, A, G_2$, $B \leftarrow B_1, \ldots, B_n$ is a fresh renaming of a clause in P, $E_1 \cup E_{A,B}$ is solvable, $\langle P, P_{\mathsf{co}} \rangle; S \cup \{A\} \Vdash \langle B_1, \ldots, B_n; E_1 \cup E_{A,B} \rangle \Rrightarrow E_3$ and $\langle P, P_{\mathsf{co}} \rangle; S \Vdash \langle G_1, G_2; E_3 \rangle \Rrightarrow E_2$ hold. We can assume that variables occurring in the selected clause do not belong to $\mathsf{dom}(\sigma_1)$ since such variables are fresh and $\mathsf{dom}(\sigma_1)$ is a finite set. Since $E_1 \cup E_{A,B} \subseteq E_3 \subseteq E_2$ by Proposition 9.2, we have $\theta_2 \in \mathsf{sol}(E_1 \cup E_{A,B})$ and denote by θ_1' the restriction of θ_2 to $\mathsf{dom}(\theta_1) \cup \mathsf{Var}(B)$. It is easy to see that, by construction, $\theta_1 \preceq \theta_1'$ and $\theta_1' \in \mathsf{sol}(E_1 \cup E_{A,B})$ and $\theta_1' \| \sigma_1$, since $\mathsf{dom}(\theta_1') \cap \mathsf{dom}(\sigma_1) = \mathsf{dom}(\theta_1)$ and $\theta_1 \preceq \sigma_1$ by hypothesis. Hence, $\sigma_1' = \theta_1' \uplus \sigma_1$ is well-defined and $\theta_1' \preceq \sigma_1'$. Since $E_3 \subseteq E_2$, $\theta_2 \in \mathsf{sol}(E_3)$ and, if θ_2' is the restriction of θ_2 to $\mathsf{dom}(\theta_1') \cup \mathsf{Var}(E_3)$, then $\theta_2' \in \mathsf{sol}(E_3)$ as well, and $\theta_1' \preceq \theta_2'$. Therefore, by induction hypothesis, we get that $\langle P, P_{\mathsf{co}} \rangle; S \cup \{A\} \Vdash \langle B_1, \ldots, B_n; E_1' \cup E_{A,B} \rangle \Rrightarrow E_3'$ holds and there is $\sigma_2' \in \mathsf{sol}(E_3')$ such that $\theta_2' \preceq \sigma_2'$, with $E_3 \subseteq E_3'$. Since $\theta_2' \preceq \theta_2$ and $\theta_2' \preceq \sigma_2'$, again by induction hypothesis, we get that $\langle P, P_{\mathsf{co}} \rangle; S \Vdash \langle G_1, G_2; E_3' \rangle \Rrightarrow E_2'$ holds and there is $\sigma_2 \in \mathsf{sol}(E_2')$ such that $\theta_2 \preceq \sigma_2$, with $E_2 \subseteq E_2'$. Then, the thesis follows by applying rule (STEP).

Case: (CO-HYP) We know that $G = G_1, A, G_2$, $E_1 \cup E_{A,B}$ is solvable for some $B \in S$, $\langle P \cup P_{\mathsf{co}}, \emptyset \rangle; \emptyset \Vdash \langle A; E_1 \cup E_{A,B} \rangle \Rrightarrow E_3$ and $\langle P, P_{\mathsf{co}} \rangle; S \Vdash \langle G_1, G_2; E_3 \rangle \Rrightarrow E_2$ hold. Since $E_1 \cup E_{A,B} \subseteq E_3 \subseteq E_2$ by Proposition 9.2, we have $\theta_2 \in \mathsf{sol}(E_1 \cup E_{A,B})$ and denote by θ_1' the restriction of θ_2 to $\mathsf{dom}(\theta_1) \cup \mathsf{Var}(B)$, but, as $\mathsf{Var}(B) \subseteq \mathsf{Var}(E_1)$, $\theta_1 \in \mathsf{sol}(E_1)$ and $\theta_1 \preceq \theta_2$, we get $\theta_1' = \theta_1$, thus $\theta_1 \in \mathsf{sol}(E_1 \cup E_{A,B})$. Hence, since $\theta_1 \preceq \sigma_1$, we get $\sigma_1 \in \mathsf{sol}(E_1 \cup E_{A,B})$ and so it also belongs to $\mathsf{sol}(E_1' \cup E_{A,B})$, that is, in particular, $E_1' \cup E_{A,B}$ is solvable. Since $E_3 \subseteq E_2$, $\theta_2 \in \mathsf{sol}(E_3)$ and, if θ_2' is the restriction of θ_2 to $\mathsf{dom}(\theta_1) \cup \mathsf{Var}(E_3)$, then $\theta_2' \in \mathsf{sol}(E_3)$ as well, and $\theta_1 \preceq \theta_2'$. Therefore, by induction hypothesis, we get $\langle P \cup P_{\mathsf{co}}, \emptyset \rangle; \emptyset \Vdash \langle A; E_1' \cup E_{A,B} \rangle \Rrightarrow E_3'$ and there is $\sigma_2' \in \mathsf{sol}(E_3')$ such that $\theta_2' \preceq \sigma_2'$, with $E_3 \subseteq E_3'$. Since $\theta_2' \preceq \theta_2$

and $\theta_2' \preceq \sigma_2'$, again by induction hypothesis, we get that $\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle G_1, G_2; E_3'\rangle \Rightarrow E_2'$ holds and there is $\sigma_2 \in \mathsf{sol}(E_2')$ such that $\theta_2 \preceq \sigma_2$, with $E_2 \subseteq E_2'$. Then, the thesis follows by applying rule (CO-HYP).

□

LEMMA 9.14 : Let $\langle G; E\rangle$ be a goal, $\theta \in \mathsf{sol}(E)$ and E_1 and E_2 be sets of equations such that $\theta \preceq \theta_1$ and $\theta \preceq \theta_2$, for some $\theta_1 \in \mathsf{sol}(E_1)$ and $\theta_2 \in \mathsf{sol}(E_2)$. If $\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle A; E\rangle \Rightarrow E_1$ and $\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle G_1, G_2; E\rangle \Rightarrow E_2$, then there exists E_3 such that $\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle G_1, A, G_2; E\rangle \Rightarrow E_3$ and $\theta \preceq \theta_3$, for some $\theta_3 \in \mathsf{sol}(E_3)$.

Proof: We sketch the proof. By Proposition 9.2, we have $E \subseteq E_1$ and by Proposition 9.13 we get $\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle G_1, G_2; E_1\rangle \Rightarrow E_3$, with $E_2 \subseteq E_3$ and $\theta_2 \preceq \theta_3$, for some $\theta_3 \in \mathsf{sol}(E_3)$. By transitivity of $\preceq$ we get $\theta \preceq \theta_3$. Then, the thesis follows by case analysis on the last applied rule in the derivation of $\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle A; E\rangle \Rightarrow E_1$, by replacing the premise $\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle \varepsilon; E_1\rangle \Rightarrow E_1$ with the judgement $\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle G_1, G_2; E_1\rangle \Rightarrow E_3$. □

LEMMA 9.15 : Let A be an atom and E be a set of equations. For all $\theta \in \mathsf{sol}(E)$, if $A\theta \in \mu[\![P \cup P_{\mathsf{co}}]\!]$, then there exists E' such that $\langle P \cup P_{\mathsf{co}}, \emptyset\rangle; \emptyset \Vdash \langle A; E\rangle \Rightarrow E'$ and $\theta \preceq \sigma$, for some $\sigma \in \mathsf{sol}(E')$.

Proof: The proof is by induction on the derivation of $A\theta$ in $\|P \cup P_{\mathsf{co}}\|$. Let $A' \leftarrow A_1, \ldots, A_n \in \|P \cup P_{\mathsf{co}}\|$ be the last applied rule in the finite derivation of $A\theta$, hence we have $A' = A\theta$. By definition of $\|P \cup P_{\mathsf{co}}\|$, we know there is a fresh renaming of a clause in $P \cup P_{\mathsf{co}}$, denote it by $B \leftarrow B_1, \ldots, B_n$, and a substitution θ' such that $B\theta' = A'$ and $B_i\theta' = A_i$, for all $i \in 1..n$. Since the variables in this clause are fresh, we can assume $\mathsf{dom}(\theta) \cap \mathsf{dom}(\theta') = \emptyset$, hence $\theta'' = \theta \uplus \theta'$ is well-defined and, by construction, we have $\theta \preceq \theta''$, thus $\theta'' \in \mathsf{sol}(E)$, and $A\theta'' = B\theta''$, that is, $\theta'' \in \mathsf{sol}(E_{A,B})$. As a consequence θ'' is a solution of $E \cup E_{A,B}$, hence, by induction hypothesis, we get that, for all $i \in 1..n$, $\langle P \cup P_{\mathsf{co}}, \emptyset\rangle; \emptyset \Vdash \langle B_i; E \cup E_{A,B}\rangle \Rightarrow E_i$ holds and $\theta'' \preceq \sigma_i$, for some $\sigma_i \in \mathsf{sol}(E_i)$.

By applying n times Lemma 9.14, we get $\langle P \cup P_{\mathsf{co}}, \emptyset\rangle; \emptyset \Vdash \langle B_1, \ldots, B_n; E \cup E_{A,B}\rangle \Rightarrow E'$ and $\theta'' \preceq \sigma$, for some $\sigma \in \mathsf{sol}(E')$. Then, the thesis follows by applying rule (STEP) to this judgement and $\langle P \cup P_{\mathsf{co}}, \emptyset\rangle; \emptyset \Vdash \langle \varepsilon; E'\rangle \Rightarrow E'$. □

LEMMA 9.16 : For all S and $\langle A; E\rangle$,
if $P^{\cup P_{\mathsf{co}}} \vdash_\mu S\theta \triangleright A\theta$, and $\theta \in \mathsf{sol}(E)$, then $\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle A; E\rangle \Rightarrow E'$ and $\theta \preceq \sigma$, for some E' and $\sigma \in \mathsf{sol}(E')$.

Proof: The proof is by induction on the derivation of $S\theta \triangleright A\theta$ (see Definition 9.8).

Case: (B-HP) We know that $A\theta \in S\theta$ and $A\theta \in \mu[\![P \cup P_{\mathsf{co}}]\!]$, that is, $\theta \in \mathsf{ans}_{\mathsf{bd}}(A, E)$. By Lemma 9.15, we get $\langle P \cup P_{\mathsf{co}}, \emptyset\rangle; \emptyset \Vdash \langle A; E\rangle \Rightarrow E_1$ and $\theta \preceq \theta_1$, for some $\theta_1 \in \mathsf{sol}(E_1)$. Since $A\theta \in S\theta$, we know that there is $B \in S$

such that $A\theta = B\theta$, that is $\theta \in \mathsf{sol}(E_{A,B})$, thus $\theta \in \mathsf{sol}(E \cup E_{A,B})$. Therefore, by Proposition 9.13, we get $\langle P \cup P_{\mathsf{co}}, \emptyset\rangle; \emptyset \Vdash \langle A; E \cup E_{A,B}\rangle \Rrightarrow E'$ and $\theta \preceq \theta_1 \preceq \sigma$, for some $\sigma \in \mathsf{sol}(E')$. The thesis follows by applying rule (CO-HYP) to this judgement and $\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle \varepsilon; E'\rangle \Rrightarrow E'$.

Case: (B-UNFOLD) We know there is a $A' \leftarrow A_1, \ldots, A_n \in \|P\|$ such that $A' = A\theta$ and $P^{\bigcirc P_{\mathsf{co}}} \vdash_\mu S\theta \cup \{A\theta\} \triangleright A_i$ is derivable, for all $i \in 1..n$. By definition of $\|P \cup P_{\mathsf{co}}\|$, we know there is a fresh renaming of a clause in $P \cup P_{\mathsf{co}}$, denote it by $B \leftarrow B_1, \ldots, B_n$, and a substitution θ' such that $B\theta' = A'$ and $B_i\theta' = A_i$, for all $i \in 1..n$. Since the variables in this clause are fresh, we can assume $\mathsf{dom}(\theta) \cap \mathsf{dom}(\theta') = \emptyset$, hence $\theta'' = \theta \uplus \theta'$ is well-defined and, by construction, we have $\theta \preceq \theta''$, thus $\theta'' \in \mathsf{sol}(E)$, and $A\theta'' = B\theta''$, that is, $\theta'' \in \mathsf{sol}(E_{A,B})$ and $S\theta \cup \{A\theta\} = (S \cup \{A\})\theta''$. As a consequence θ'' is a solution of $E \cup E_{A,B}$, hence, by induction hypothesis, we get that, for all $i \in 1..n$, $\langle P, P_{\mathsf{co}}\rangle; S \cup \{A\} \Vdash \langle B_i; E \cup E_{A,B}\rangle \Rrightarrow E_i$ holds and $\theta'' \preceq \sigma_i$, for some $\sigma_i \in \mathsf{sol}(E_i)$.
By applying n times Lemma 9.14, we get $\langle P, P_{\mathsf{co}}\rangle; S \cup \{A\} \Vdash \langle B_1, \ldots, B_n; E \cup E_{A,B}\rangle \Rrightarrow E'$ and $\theta'' \preceq \sigma$, for some $\sigma \in \mathsf{sol}(E')$. By transitivity we get $\theta \preceq \sigma$. Then, the thesis follows by applying rule (STEP) to this judgement and $\langle P \cup P_{\mathsf{co}}, \emptyset\rangle; \emptyset \Vdash \langle \varepsilon; E'\rangle \Rrightarrow E'$.

□

Proof (Lemma 9.11)*:* The proof is by induction on the number of atoms in $G = A_1, \ldots, A_n$. If $G = \varepsilon$, then the thesis follows by rule (EMPTY), taking $E' = E$ and $\sigma = \theta$. If $G = G_1, A, G_2$, we know by hypothesis that $P^{\bigcirc P_{\mathsf{co}}} \vdash_\mu S\theta \triangleright A\theta$, hence, by Lemma 9.16 we get $\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle A; E\rangle \Rrightarrow E_1$ and $\theta \preceq \theta_1$, for some $\theta_1 \in \mathsf{sol}(E_1)$. By induction hypothesis, we get $\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle G_1, G_2; E\rangle \Rrightarrow E_2$ and $\theta \preceq \theta_2$, for some $\theta_2 \in \mathsf{sol}(E_2)$, therefore, by Lemma 9.14, we get $\langle P, P_{\mathsf{co}}\rangle; S \Vdash \langle G_1, A, G_2; E\rangle \Rrightarrow E'$ and $\theta \preceq \sigma$ for some $\sigma \in \mathsf{sol}(E')$. □

10

Discussion

Regular coinduction is an interesting compromise between induction and coinduction, combining advantages of both approaches: it is not restricted to finite derivations, thus overcoming limitations of the inductive approach, but it still has a finite nature, as a regular derivation can be non-well-founded, but it can only contain finitely many judgements. Hence, regular coinduction is a simple alternative, which combines the flexibility of non-well-founded derivations with the possibility of having sound and complete procedures that find a derivation, if any. Indeed, many concrete examples supporting coinduction, from proof and type systems to programming languages, actually support regular coinduction, thus it is important to provide solid foundations also to this case.

We address this task in Chapter 8. We extend the analysis described in Chapter 2 for standard interpretations to the regular case. More in detail, we start from the natural proof-theoretic definition of the regular interpretation of an inference system, as the set of judgements derivable by a regular proof tree. To provide a fixed point characterisation of this interpretation, we rely on the rational fixed point construction by Adámek, Milius, and Velebil (2006), restricted to the lattice-theoretic setting. Indeed, the regular interpretation turns out to coincide with the rational fixed point of the inference operator. Then, we show that the regular interpretation has an equivalent inductive characterization, which provides us with an algorithm to find a derivation for a judgment, if any. Relying on these results, we discuss proof techniques for regular reasoning: from the fixed point characterisation we got the regular coinduction principle, which allows us to prove completeness, while from the inductive characterization we derived a proof technique to show soundness.

Regular coinduction suffers from the same limitations of standard coinduction described in Chapter 3, which inference systems with corules overcome. Hence, we extend all results presented for regular coinduction to this generalised framework, thus providing a flexible approach also to regular reasoning.

Finally, in Chapter 9, we consider a case where regular coinduction plays a crucial role: coinductive logic programming (Simon, 2006; Simon et al., 2006, 2007). We provide a detailed formal account of an extension of logic programming where programs are enriched by coclauses, the counterpart of corules, which can be used to tune the interpretation of predicates. Viewing logic programs as a particular, syntactic, instance of inference systems, we define a declarative semantics of logic programs with coclauses the same way

as inference systems with corules, that is, as the largest comodel below a certain set determined by coclauses. Then, we define an operational semantics, as a combination of standard SLD resolution and coSLD resolution, which, being inductively defined and managing only finite objects, provides a semi-algorithm to solve a goal.

The proposed operational semantics is not incidental, but is the logic programming counterpart of the inductive characterisation of the regular interpretation of an inference system. Following this analogy, we define the regular declarative semantics of a logic program with coclauses (the union of all finite comodels below a certain set determined by coclauses) and, relying on results proved for inference systems, we show that the proposed operational semantics is sound and complete with respect to such regular declarative semantics.

It has been shown by Ancona and Dovier (2015) that, taking the coinductive declarative semantics (the largest comodel), there is not even a semi-algorithm, which is complete and checks that an atom belongs to that semantics. Hence, there is no hope to find a complete operational semantics in general. The same result applies also to our first declarative semantics, as it generalises the coinductive one. On the other hand, our results provide, for an extension of logic programming, fully-developed foundations and results which are exactly the analogous of those for standard logic programming.

10.1 Related work

The regular approach has been adopted in many different contexts, notably to define proof systems for several kinds of logics, and to define operational models of programming languages supporting cyclic structures.

Concerning proof systems supporting regular proofs, we find proposals by Santocanale (2002), Fortier and Santocanale (2013), and Doumane (2017) for logics with fixed point operators, and by Brotherston (2005) and Brotherston and Simpson (2011) for classical first order logic with inductive definitions. In both cases, regular proofs allow to naturally handle the unfolding of fixed point and recursive definitions, respectively. However, regular proofs allow the derivation of wrong sequents, such as the empty one; hence, to solve this issue, they have to impose additional constraints on regular proofs, to disregard incorrect derivations. These additional requirements on regular proofs are expressed at the meta-level and typically require some condition to hold infinitely often in the regular proof. As inference systems with corules have been designed precisely to filter out undesired infinite derivations, and they seem pretty good to capture requirements that should hold infinitely often in the proof, it would be interesting to investigate whether these additional constraints can be enforced by an appropriate set of corules.

The other context where we can find applications of regular coinduction is in programming languages supporting cyclic structures. In this case, we use the term *regular corecursion* for a semantics of recursive definitions which detects cycles, analogously to the inductive characterization of the regular interpreta-

tion. We can find proposals of language constructs for regular corecursion in all common programming paradigms: logic paradigm, by Simon (2006), Simon et al. (2006, 2007), and Ancona and Dovier (2015), functional paradigm, by Jeannin, Kozen, and Silva, 2013, 2017 and object-oriented paradigm, by Ancona and Zucca (2012).

Coinductive logic programming (CoLP) has been initially proposed by Simon (2006) and Simon et al. (2006) as a convenient sub-paradigm of logic programming able to deal with circularity. Its limitations, inherited from the coinductive interpretation of clauses, were soon recognized and Simon et al. (2007) proposed a more complex model where inductive and coinductive predicates can coexist in the same program, provided that they can be stratified.

Moura (2013), Mantadelis, Rocha, and Moura (2014), and Ancona (2013) have proposed implementations of coLP based on refinements of the Simon's original proposal, with the main aim of making them more portable and flexible. Ancona (2013) has extended coLP by introducing a `finally` clause, allowing the user to define the specific behavior of a predicate when solved by coinductive hypotheses. Implementation by Moura (2013) and Mantadelis, Rocha, and Moura (2014) is embedded in a tabled Prolog related to the implementation of Logtalk, and is based on a mechanism similar to `finally` clauses to specify customized behavior of predicates when solved by coinductive hypotheses. While such mechanisms resemble coclauses, the corresponding formalization is purely operational and lacks a declarative semantics and corresponding proof principles for proving correctness of predicate definitions based on them.

Ancona and Dovier (2015) have proposed an operational semantics of coLP based on the big-step approach, which is simpler than the operational semantics initially proposed by Simon et al. (2006) and proved it to be sound. They have also formally shown that there is no complete procedure for deciding whether a regular goal belongs to the coinductive declarative semantics, but provided neither completeness result restricted to regular derivations, nor mechanisms to extend coLP and make it more flexible.

While coSLD resolution and its proposed extensions are limited by the fact that cycles must be detected in derivations to allow resolution to succeed, a stream of work based on the notion of *structural resolution* (Johann, Komendantskaya, and Komendantskiy, 2015; Komendantskaya, Power, and Schmidt, 2016; Komendantskaya, Johann, and Schmidt, 2016) (S-resolution for short) aims to make coinductive resolution more powerful, by allowing to lazily detect infinite derivations which do not have cycles. In particular, recent results by Li (2017), Komendantskaya and Li (2017), and Basold, Komendantskaya, and Li (2019) investigate how it is possible to integrate coLP cycle detection into S-resolution, by proposing a comprehensive theory. We plan to investigate in future work how to integrate coclauses and structural resolution, to combine advantages of both paradigms.

In the object-oriented paradigm, coFJ (Ancona and Zucca, 2012) is an extension of FJ (Igarashi, Pierce, and Wadler, 2001) supporting regular objects and regular corecusion. As in coLP, regular objects are represented by syn-

tactic term equations and the evaluation keeps tack of already encountered method calls, so that it can detect cycles. Further, the programmer can specify some code (called codefinition) to be executed when cycles are detected. This mechanism is very similar to corules and, acutally, it was the starting point for the development of this framework, which was initially meant for providing a more abstract semantics to this kind of language constructs. Once introduced inference systems with corules, we realised that the operational model by Ancona and Zucca (2012) allows the derivation of spurious results in some cases. To solve this issue, we recently proposed (Barbieri et al., 2019; Ancona et al., 2020b; Barbieri, Dagnino, and Zucca, 2020) a more principled approach to coFJ semantics: we provided an abstract semantics based on inference system with corules, serving as reference model, an inductive abstract semantics, modelling the cycle detection mechanism independently from the representation of infinite objects, and an operational semantics, which chooses a concrete representation of values and serves as a guide to develop an interpreter.

CoCaml[1] proposed by Jeannin, Kozen, and Silva (2017) and Jeannin and Kozen (2012) is a fully-fledged extension of OCaml supporting regular non-well-founded data types and corecursive functions. CoCaml, as OCaml, allows programmers to declare regular values through the let-rec construct, and, moreover, detects cyclic calls. However, the CoCaml approach is in two phases. First, a system of equations is constructed, associating with each call a variable and partially evaluating the body of functions, where calls are replaced with associated variables. Then, the system of equations is given to a *solver* specified in the function definition. Solvers can be either pre-defined or written by the programmer in order to enhance flexibility. An advantage we see in our approaches based on corules (flexible coLP and coFJ) is that the programmer has to write coclauses and codefinitions, which are standard code, rather than working at the meta-level to write a solver, which is in a sense a fragment of the interpreter.

10.2 Future work

An interesting direction is the development of more sophisticated proof techniques for regular reasoning. Indeed, several enhanced coinductive techniques have been proposed, such as parametrized coinduction (Hur et al., 2013) and coinduction up-to (Pous, 2007; Pous and Sangiorgi, 2012; Pous, 2016), which have been proved to be effective in several contexts. Adapting such techniques to the (flexible) regular case would provide us with powerful tools to support regular reasoning. A further development in this direction would be to provide support to regular reasoning in proof assistants, which usually provide primitives only for plain induction and coinduction. To this end, we could start from existing approaches by Spadotti (2016) and Uustalu and Veltri (2017) to implement regular terms in proof assistants.

1 Available at www.cs.cornell.edu/Projects/CoCaml

Another interesting direction is to use (flexible) regular coinduction to design in a principled way abstract and operational semantics for programming languages supporting regular corecursion in various programming paradigms. We have addressed this issue in logic programming in Chapter 9 and we have made some steps in this direction in the object-oriented paradigm, defining coFJ (Barbieri et al., 2019; Ancona et al., 2020b; Barbieri, Dagnino, and Zucca, 2020). In future work, we plan to tackle also the functional paradigm comparing our approach with CoCaml. Here the main challenge is how to detect cycles in presence of higher-order functions.

Another direction to improve the support for infinite objects in programming languages is to integrate regular corecusion with the more common approach to represent and deal with such objects in programming languages, namely, lazy evaluation. With the lazy approach, arbitrary (computable) non-well-founded objects are supported. However, we cannot compute results which need to explore the whole structure, whereas, with regular corecursion, this becomes possible for cyclic structures: for instance we can compute the maximum of a regular list, which would cause non-termination in lazy languages such as Haskell. A natural question is then whether it is possible to extend the regular corecursion approach to manage also non-regular objects, thus overcoming the principal drawback with respect to the lazy approach. A possible interesting direction, exploiting the work of Courcelle (1983) on infinite trees, could be to move from regular to *algebraic* objects.

Conclusion

11

Conclusion

In this thesis we introduce inference systems with corules, a generalisation of standard inference systems, which provides more flexibility when interpreting a given set of rules, allowing interpretations lying between the inductive and the coinductive one. The key concept of the proposed generalisation are corules, which are special rules that need to be provided together with standard rules, and are used to tune their semantics. More precisely, they allow us to refine the coinductive interpretation of standard rules, disregarding some undesired infinite derivations, thus we call this approach flexible coinduction. An important property is that standard inductive and coinductive interpretations are particular cases, that is, they can be recovered by specific choices of corules, thus this framework indeed generalises standard inference systems.

The first contribution of the thesis is to study in full detail inference systems with corules and their general properties, thus providing solid and fairly simple foundations to flexible coinduction. We extend all standard results about inference systems to this generalised setting: we define a fixed point construction that is at the basis of the model-theoretic definition of the interpretation determined by corules, provide several proof-theoretic characterisations, proving they are all equivalent to the model-theoretic one, and describe proof techniques to reason with corules.

The second contribution is the analysis of the paradigmatic example of big-step operational semantics modelling also infinite behaviour, where neither the inductive nor the coinductive interpretation of rules are able to capture the intended meaning, showing how corules can be successfully adopted to solve this problem. We consider first semantic descriptions where the behaviour of the program is just described by its final result, if any, and a special result is used to model divergence. Then, we extend the approach to more complex descriptions where, in addition to the final result, we also have observations, modelling the interaction with the environment (*e.g.*, traces of events, memory usage, costs etc.). In this latter case, considering also infinite behaviour is even more challenging, as we need to model possibly infinite observable interactions. The key contribution is that, rather than studying big-step semantics on example languages, we take a general perspective, developing our definitions and results for an arbitrary big-step semantics, abstracting from specific features of concrete examples. To this end, we provide a definition of what is a big-step semantics with or without observations, we then define computations by means of a transition relation driven by rules modelling the evaluation

algorithm implicitly associated with a big-step specification and serving as a reference model, and, finally, we provide constructions producing a big-step semantics able to distinguish between stuck and infinite computations. Furthermore, in the former case without observations, relying on the proposed constructions, we show how to express and prove soundness of a predicate against a big-step semantics, providing a proof technique based on sufficient conditions on individual big-step rules. The generality of our approach is witnessed by a broad class of examples.

The third contribution is the study of algorithmic variants of the general framework of inference systems with corules, to provide concrete support to them. To this end, we consider the restriction of the general model to regular derivations, as it is customary in standard coinduction. We extend all notions and results discussed in the general setting to the regular one, thus providing solid foundations also to flexible regular coinduction. From the algorithmic perspective, the important result is that flexible regular coinduction has an equivalent inductive characterisation, which provides us with a sound and complete (abstract) algorithm to find a derivation. Building on this general analysis, we define an extension of logic programming supporting flexible coinduction restricted to the regular case, like standard coinductive logic programming. We define both declarative and operational semantics of flexible coinductive logic programming, showing the latter is equivalent to the regular restriction of the former one, and we also have a prototype SWI-Prolog implementation available at https://github.com/davideancona/coLP-with-coclauses.

The notion of corules has been inspired by some operational models for programming languages supporting corecursion, *e.g.*, those proposed by Ancona and Zucca (2012, 2013) and Ancona (2013). The original aim was to define a more abstract counterpart of these models, enabling formal reasoning on them. Thanks to this abstract reference model, we realised that in some cases operational semantics proposed in these works provided incorrect results, hence we started revising them following the abstract model described by corules. In this thesis we have addressed the logic paradigm, as it is the closest to the abstract model. We have also worked on the object-oriented paradigm (Barbieri et al., 2019; Ancona et al., 2020b; Barbieri, Dagnino, and Zucca, 2020). These results have been not included because they have specific issues to be faced, which deviate from the main focus of this thesis, notably the fact that we have to deal with functions rather than predicates.

11.1 Future work

We have already discussed future work for each part in Sections 4.2, 7.2 and 10.2. Here we just recall main lines for future developments.

ABSTRACT MODEL It would be interesting to investigate variants of the interpretation of an inference system with corules, to avoid the unexpected behaviours that sometimes may happen. To this end, we could extend the framework to take into account the definition of multiple judgements

at the same time, or change the definition of the model to make stronger the connection between the two steps of the definition.

PROOF METHODS Proof techniques to reason with corules are very important. In this direction we plan to extend well-known techniques for standard coinduction, such as up-to techniques (Pous, 2007; Pous and Sangiorgi, 2012; Pous, 2016) or parametric coinduction (Hur et al., 2013), to our generalised setting, also in the regular case. Further, another important direction is to provide mechanised support to corules, by implementing them in a proof assistant such as Agda or Coq.

BIG-STEP SEMANTICS In this setting, it would be interesting to use our general framework to define and reason about other constructions extending big-step semantics to include infinite behaviour, and also to define other proof techniques to prove relevant semantic properties, such as soundness or normalisation. Furthermore, we also plan to extend our approach to take into account other computational models, such as probabilistic computations.

PROGRAMMING LANGUAGES Providing support to flexible coinduction in programming languages is a challenging direction. We have alredy done some work on logic and object-oriented paradigm. In future work we plan to address also the functional paradigm, where the main issue is to deal with higher-order functions. Furthermore, we plan to extend the current approach, restricted to regular case, by combining it with existing techniques based on lazy evaluation.

Bibliography

Abadi, Martín and Luca Cardelli (1996). *A Theory of Objects*. Monographs in Computer Science. Springer. DOI: 10.1007/978-1-4419-8598-9. Cited on p. 86.

Abel, Andreas and James Chapman (2014). Normalization by Evaluation in the Delay Monad: A Case Study for Coinduction via Copatterns and Sized Types. In: *Proceedings 5th Workshop on Mathematically Structured Functional Programming, MSFP@ETAPTS 2014*. Edited by Paul Levy and Neel Krishnaswami. Vol. 153. Electronic Proceedings in Theoretical Computer Science, pp. 51–67. DOI: 10.4204/EPTCS.153.4. Cited on p. 171.

Abel, Andreas and Brigitte Pientka (2013). Wellfounded recursion with copatterns: a unified approach to termination and productivity. In: *Proceedings of the 18th ACM International Conference on Functional Programming, ICFP 2013*. Edited by Greg Morrisett and Tarmo Uustalu. ACM Press, pp. 185–196. DOI: 10.1145/2500365.2500591. Cited on p. 63.

Abel, Andreas, Brigitte Pientka, David Thibodeau, and Anton Setzer (2013). Copatterns: programming infinite structures by observations. In: *The 40th Annual ACM Symposium on Principles of Programming Languages, POPL'13*. Edited by Roberto Giacobazzi and Radhia Cousot. ACM Press, pp. 27–38. DOI: 10.1145/2429069.2429075. Cited on p. 63.

Aczel, Peter (1977). An Introduction to Inductive Definitions. In: *Handbook of Mathematical Logic*. Edited by Jon Barwise. Vol. 90. Studies in Logic and the Foundations of Mathematics. Elsevier, pp. 739 –782. Cited on pp. 11, 12, 62.

Aczel, Peter, Jirí Adámek, and Jiri Velebil (2001). A Coalgebraic View of Infinite Trees and Iteration. In: *Electronic Notes in Theoretical Computer Science* 44.1, pp. 1–26. DOI: 10.1016/S1571-0661(04)80900-9. Cited on p. 13.

Aczel, Peter, Jirí Adámek, Stefan Milius, and Jiri Velebil (2003). Infinite trees and completely iterative theories: a coalgebraic view. In: *Theoretical Computer Science* 300.1-3, pp. 1–45. DOI: 10.1016/S0304-3975(02)00728-4. Cited on pp. 13–15.

Adámek, Jirí, Stefan Milius, and Jiri Velebil (2006). Iterative algebras at work. In: *Mathematical Structures in Computer Scienc* 16.6, pp. 1085–1131. DOI: 10.1017/S0960129506005706. Cited on pp. 178, 180, 181, 221.

Adámek, Jirí, Paul Blain Levy, Stefan Milius, Lawrence S. Moss, and Lurdes Sousa (2015). On Final Coalgebras of Power-Set Functors and Saturated Trees. In: *Applied Categorical Structures* 23.4, pp. 609–641. DOI: 10.1007/s10485-014-9372-9. Cited on pp. 13–15.

Ager, Mads Sig (2004). From Natural Semantics to Abstract Machines. In: *Logic-Based Program Synthesis and Transformation - 14th International Symposium, LOPSTR 2004*. Edited by Sandro Etalle. Vol. 3573. Lecture Notes in Computer Science. Springer, pp. 245–261. DOI: 10.1007/11506676_16. Cited on p. 169.

Amin, Nada and Tiark Rompf (2017). Type Soundness Proofs with Definitional Interpreters. In: *The 44th Annual ACM Symposium on Principles of Program-*

ming Languages, POPL'17. Edited by Giuseppe Castagna and Andrew D. Gordon. ACM Press, pp. 666–679. DOI: 10.1145/3009837. Cited on pp. 169, 170, 172.

Ancona, Davide (2012). Soundness of Object-Oriented Languages with Coinductive Big-Step Semantics. In: *26th European Conference on Object-Oriented Programming, ECOOP 2012*. Edited by James Noble. Vol. 7313. Lecture Notes in Computer Science. Springer, pp. 459–483. DOI: 10.1007/978-3-642-31057-7_21. Cited on pp. 62, 121.

Ancona, Davide (2013). Regular corecursion in Prolog. In: *Computer Languages, Systems & Structures* 39.4, pp. 142–162. DOI: 10.1016/j.cl.2013.05.001. Cited on pp. 33, 57, 62, 203, 223, 230.

Ancona, Davide (2014). How to Prove Type Soundness of Java-like Languages without Forgoing Big-Step Semantics. In: *Proceedings of 16th Workshop on Formal Techniques for Java-like Programs, FTfJP'14*. Edited by David J. Pearce. ACM Press, 1:1–1:6. DOI: 10.1145/2635631.2635846. Cited on pp. 62, 121, 170.

Ancona, Davide and Andrea Corradi (2014). Sound and Complete Subtyping between Coinductive Types for Object-Oriented Languages. In: *28th European Conference on Object-Oriented Programming, ECOOP 2014*. Edited by Richard E. Jones. Vol. 8586. Lecture Notes in Computer Science. Springer, pp. 282–307. DOI: 10.1007/978-3-662-44202-9_12. Cited on p. 62.

Ancona, Davide, Francesco Dagnino, and Elena Zucca (2017a). Extending Coinductive Logic Programming with Co-Facts. In: *Proceedings of the First Workshop on Coalgebra, Horn Clause Logic Programming and Types, CoALP-Ty'16*. Edited by Ekaterina Komendantskaya and John Power. Vol. 258. Electronic Proceedings in Theoretical Computer Science. Open Publishing Association, pp. 1–18. DOI: 10.4204/EPTCS.258.1. Cited on p. 8.

Ancona, Davide, Francesco Dagnino, and Elena Zucca (2017b). Generalizing Inference Systems by Coaxioms. In: *Programming Languages and Systems - 26th European Symposium on Programming, ESOP 2017*. Edited by Hongseok Yang. Vol. 10201. Lecture Notes in Computer Science. Springer, pp. 29–55. DOI: 10.1007/978-3-662-54434-1_2. Cited on pp. 7, 33.

Ancona, Davide, Francesco Dagnino, and Elena Zucca (2017c). Reasoning on Divergent Computations with Coaxioms. In: *Proceedings of ACM on Programming Languages* 1.OOPSLA, 81:1–81:26. DOI: 10.1145/3133905. Cited on pp. 7, 121, 168, 169.

Ancona, Davide, Francesco Dagnino, and Elena Zucca (2018). Modeling Infinite Behaviour by Corules. In: *32nd European Conference on Object-Oriented Programming, ECOOP 2018*. Edited by Todd D. Millstein. Vol. 109. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 21:1–21:31. DOI: 10.4230/LIPIcs.ECOOP.2018.21. Cited on pp. 7, 121, 122, 148, 168, 169.

Ancona, Davide and Agostino Dovier (2015). A Theoretical Perspective of Coinductive Logic Programming. In: *Fundamenta Informaticae* 140.3-4, pp. 221–246. DOI: 10.3233/FI-2015-1252. Cited on pp. 64, 178, 205, 208, 214, 222, 223.

Ancona, Davide and Giovanni Lagorio (2009). Coinductive type systems for object-oriented languages. In: *23rd European Conference on Object-Oriented Programming, ECOOP 2009*. Edited by Sophia Drossopoulou. Vol. 5653. Lecture Notes in Computer Science. Springer, pp. 2–26. DOI: 10.1007/978-3-642-03013-0. Cited on p. 62.

Ancona, Davide and Elena Zucca (2012). Corecursive Featherweight Java. In: *Proceedings of 14th Workshop on Formal Techniques for Java-like Programs, FTfJP'12*. Edited by Wei-Ngan Chin and Aquinas Hobor. ACM Press, pp. 3–10. DOI: 10.1145/2318202. Cited on pp. 33, 62, 223, 224, 230.

Ancona, Davide and Elena Zucca (2013). Safe corecursion in coFJ. In: *Proceedings of 15th Workshop on Formal Techniques for Java-like Programs, FTfJP'13*. Edited by Werner Dietl. ACM Press, 2:1–2:7. DOI: 10.1145/2489804. Cited on pp. 33, 62, 230.

Ancona, Davide, Francesco Dagnino, Jurriaan Rot, and Elena Zucca (2020a). A big step from finite to infinite computations. In: *Science of Computer Programming* 197, p. 102492. DOI: 10.1016/j.scico.2020.102492. Cited on pp. 7, 172.

Ancona, Davide, Pietro Barbieri, Francesco Dagnino, and Elena Zucca (2020b). Sound regular corecursion in coFJ. In: *34th European Conference on Object-Oriented Programming, ECOOP 2020*. Vol. 166. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. Cited on pp. 8, 33, 224, 225, 230.

Apt, Krzysztof R. (1997). *From logic programming to Prolog*. Prentice Hall International series in computer science. Prentice Hall. Cited on pp. 204, 205, 208.

Arnold, André and Maurice Nivat (1980). The metric space of infinite trees. Algebraic and topological properties. In: *Fundamenta Informaticae* 3.4, pp. 445–476. Cited on p. 47.

Barbanera, Franco, Mariangiola Dezani-Ciancaglini, and Ugo de'Liguoro (1995). Intersection and Union Types: Syntax and Semantics. In: *Information and Computation* 119.2, pp. 202–230. DOI: 10.1006/inco.1995.1086. Cited on p. 112.

Barbieri, Pietro, Francesco Dagnino, and Elena Zucca (2020). An inductive abstract semantics for coFJ. In: *Proceedings of 22nd Workshop on Formal Techniques for Java-like Programs, FTfJP'20*. ACM Press. Cited on pp. 8, 224, 225, 230.

Barbieri, Pietro, Francesco Dagnino, Elena Zucca, and Davide Ancona (2019). Corecursive Featherweight Java Revisited. In: *Proceedings of the 20th Italian Conference on Theoretical Computer Science, ICTCS 2019*. Edited by Alessandra Cherubini, Nicoletta Sabadini, and Simone Tini. Vol. 2504. CEUR Workshop Proceedings. CEUR-WS.org, pp. 158–170. Cited on pp. 8, 224, 225, 230.

Barendregt, Hendrik Pieter, Wil Dekkers, and Richard Statman (2013). *Lambda Calculus with Types*. Perspectives in logic. Cambridge University Press. Cited on p. 110.

Basold, Henning (2018). *Mixed Inductive-Coinductive Reasoning: Types, Programs and Logic*. PhD thesis. Radboud University Nijmegen. Cited on p. 63.

Basold, Henning and Ekaterina Komendantskaya (2016). Models of Inductive-Coinductive Logic Programs. In: *Proceedings of the First Workshop on Coalgebra, Horn Clause Logic Programming and Types, CoALP-Ty'16*. Cited on p. 57.

Basold, Henning, Ekaterina Komendantskaya, and Yue Li (2019). Coinduction in Uniform: Foundations for Corecursive Proof Search with Horn Clauses. In: *Programming Languages and Systems - 28th European Symposium on Programming, ESOP 2019*. Edited by Luís Caires. Vol. 11423. Lecture Notes in Computer Science. Springer, pp. 783–813. DOI: 10.1007/978-3-030-17184-1_28. Cited on pp. 63, 223.

Bettini, Lorenzo, Viviana Bono, Mariangiola Dezani-Ciancaglini, Paola Giannini, and Betti Venneri (2018). Java & Lambda: a Featherweight Story. In: *Logical Methods in Computer Science* 14.3. DOI: 10.23638/LMCS-14(3:17)2018. Cited on p. 105.

Bird, Richard S. and Philip Wadler (1988). *Introduction to functional programming*. Prentice Hall International series in computer science. Prentice Hall. Cited on p. 62.

Bizjak, Ales, Hans Bugge Grathwohl, Ranald Clouston, Rasmus Ejlers Møgelberg, and Lars Birkedal (2016). Guarded Dependent Type Theory with Coinductive Types. In: *Foundations of Software Science and Computation Structures - 19th International Conference, FOSSACS 2016*. Edited by Bart Jacobs and Christof Löding. Vol. 9634. Lecture Notes in Computer Science. Springer, pp. 20–35. DOI: 10.1007/978-3-662-49630-5_2. Cited on p. 63.

Brotherston, James (2005). Cyclic Proofs for First-Order Logic with Inductive Definitions. In: *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX 2005*. Edited by Bernhard Beckert. Vol. 3702. Lecture Notes in Computer Science. Springer, pp. 78–92. DOI: 10.1007/11554554_8. Cited on pp. 63, 222.

Brotherston, James and Alex Simpson (2011). Sequent calculi for induction and infinite descent. In: *Journal of Logic and Computation* 21.6, pp. 1177–1216. DOI: 10.1093/logcom/exq052. Cited on pp. 63, 64, 222.

Capretta, Venanzio (2005). General Recursion via Coinductive Types. In: *Logical Methods in Computer Science* 1.2. DOI: 10.2168/LMCS-1(2:1)2005. Cited on p. 170.

Capretta, Venanzio, Tarmo Uustalu, and Varmo Vene (2009). Corecursive Algebras: A Study of General Structured Corecursion. In: *Formal Methods: Foundations and Applications, 12th Brazilian Symposium on Formal Methods, SBMF 2009*. Edited by Marcel Vinícius Medeiros Oliveira and Jim Woodcock. Vol. 5902. Lecture Notes in Computer Science. Springer, pp. 84–100. DOI: 10.1007/978-3-642-10452-7_7. Cited on p. 86.

Chapman, James, Tarmo Uustalu, and Niccolò Veltri (2019). Quotienting the delay monad by weak bisimilarity. In: *Mathematical Structures in Computer Scienc* 29.1, pp. 67–92. DOI: 10.1017/S0960129517000184. Cited on p. 171.

Charguéraud, Arthur (2013). Pretty-Big-Step Semantics. In: *Programming Languages and Systems - 22nd European Symposium on Programming, ESOP 2013*.

Edited by Matthias Felleisen and Philippa Gardner. Vol. 7792. Lecture Notes in Computer Science. Springer, pp. 41–60. DOI: 10.1007/978-3-642-37036-6_3. Cited on pp. 167, 168.

Ciccone, Luca (2019). *Flexible Coinduction in Agda*. MA thesis. University of Genova. Cited on p. 63.

Courcelle, Bruno (1983). Fundamental Properties of Infinite Trees. In: *Theoretical Computer Science* 25, pp. 95–169. DOI: 10.1016/0304-3975(83)90059-2. Cited on pp. 13, 14, 47, 75, 76, 178, 179, 215, 225.

Cousot, Patrick and Radhia Cousot (1977). Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In: *The 4th Annual ACM Symposium on Principles of Programming Languages, POPL'77*. Edited by Robert M. Graham, Michael A. Harrison, and Ravi Sethi. ACM, pp. 238–252. DOI: 10.1145/512950.512973. Cited on p. 23.

Cousot, Patrick and Radhia Cousot (1992). Inductive Definitions, Semantics and Abstract Interpretations. In: *The 19th Annual ACM Symposium on Principles of Programming Languages, POPL'92*. Edited by Ravi Sethi. ACM Press, pp. 83–94. DOI: 10.1145/143165.143184. Cited on pp. 62, 168.

Dagnino, Francesco (2018). Flexible Coinduction for Infinite Behaviour. In: *Proceedings of the 19th Italian Conference on Theoretical Computer Science, ICTCS 2018*. Edited by Alessandro Aldini and Marco Bernardo. Vol. 2243. CEUR Workshop Proceedings. CEUR-WS.org, pp. 17–23. Cited on p. 7.

Dagnino, Francesco (2019). Coaxioms: flexible coinductive definitions by inference systems. In: *Logical Methods in Computer Science* 15.1. DOI: 10.23638/LMCS-15(1:26)2019. Cited on pp. 7, 14, 33, 76.

Dagnino, Francesco (2020). *Foundations of regular coinduction*. Technical Report. Submitted for journal publication. DIBRIS, University of Genova. Cited on p. 8.

Dagnino, Francesco, Davide Ancona, and Elena Zucca (2020). Flexible coinductive logic programming. In: *Theory and Practice of Logic Programming* 20.6. Issue for ICLP 2020, pp. 818–833. DOI: 10.1017/S147106842000023X. Cited on pp. 8, 33.

Dagnino, Francesco, Viviana Bono, Elena Zucca, and Mariangiola Dezani-Ciancaglini (2020). Soundness Conditions for Big-Step Semantics. In: *Programming Languages and Systems - 29th European Symposium on Programming, ESOP 2020*. Edited by Peter Müller. Vol. 12075. Lecture Notes in Computer Science. Springer, pp. 169–196. DOI: 10.1007/978-3-030-44914-8_7. Cited on p. 7.

Dal Lago, Ugo and Margherita Zorzi (2012). Probabilistic operational semantics for the lambda calculus. In: *RAIRO Theoretical Informatics and Applications* 46.3, pp. 413–450. DOI: 10.1051/ita/2012012. Cited on p. 172.

Danielsson, Nils Anders (2010). Total Parser Combinators. In: *Proceedings of the 15th ACM International Conference on Functional Programming, ICFP 2010*. ACM Press, pp. 285–296. Cited on p. 170.

Danielsson, Nils Anders (2012). Operational Semantics using the Partiality Monad. In: *Proceedings of the 17th ACM International Conference on Functional Programming, ICFP 2012*. Edited by Peter Thiemann and Robby Bruce Findler. ACM Press, pp. 127–138. DOI: 10.1145/2364527.2364546. Cited on p. 170.

Davey, Brian A. and Hilary A. Priestley (2002). *Introduction to Lattices and Order*. 2nd ed. Cambridge University Press. DOI: 10.1017/CBO9780511809088. Cited on pp. 19, 26, 38.

De Nicola, Rocco and Matthew Hennessy (1984). Testing Equivalences for Processes. In: *Theoretical Computer Science* 34.1, pp. 83 –133. DOI: 10.1016/0304-3975(84)90113-0. Cited on p. 96.

Dezani-Ciancaglini, Mariangiola, Ugo de'Liguoro, and Adolfo Piperno (1998). A Filter Model for Concurrent lambda-Calculus. In: *SIAM Journal of Computing* 27.5, pp. 1376–1419. DOI: 10.1137/S0097539794275860. Cited on p. 110.

Doumane, Amina (2017). *On the infinitary proof theory of logics with fixed points. (Théorie de la démonstration infinitaire pour les logiques à points fixes)*. PhD thesis. Paris Diderot University, France. Cited on pp. 63, 222.

Fortier, Jérôme and Luigi Santocanale (2013). Cuts for circular proofs: semantics and cut-elimination. In: *Computer Science Logic 2013, CSL 2013*. Edited by Simona Ronchi Della Rocca. Vol. 23. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 248–262. DOI: 10.4230/LIPIcs.CSL.2013.248. Cited on p. 222.

Gacek, Andrew, Dale Miller, and Gopalan Nadathur (2008). Combining Generic Judgments with Recursive Definitions. In: *Proceedings of the 23rd Annual IEEE Symposium on Logic in Computer Science, LICS 2008*. IEEE Computer Society, pp. 33–44. DOI: 10.1109/LICS.2008.33. Cited on p. 63.

Gosling, James, Bill Joy, Guy L. Steele, Gilad Bracha, and Alex Buckley (2014). *The Java Language Specification, Java SE 8 Edition*. 1st. Addison-Wesley Professional. Cited on p. 106.

Grudzinski, Grzegorz (2000). A Minimal System of Disjunctive Properties for Strictness Analysis. In: *ICALP Workshops*. Edited by José D. P. Rolim, Andrei Z. Broder, Andrea Corradini, Roberto Gorrieri, Reiko Heckel, Juraj Hromkovic, Ugo Vaccaro, and J. B. Wells. Carleton Scientific, pp. 305–322. Cited on p. 110.

Gupta, Gopal, Neda Saeedloei, Brian W. DeVries, Richard Min, Kyle Marple, and Feliks Kluzniak (2011). Infinite Computation, Co-induction and Computational Logic. In: *Algebra and Coalgebra in Computer Science - 4th International Conference, CALCO 2011*. Edited by Andrea Corradini, Bartek Klin, and Corina Cîrstea. Vol. 6859. Lecture Notes in Computer Science. Springer, pp. 40–54. DOI: 10.1007/978-3-642-22944-2_4. Cited on p. 212.

Hagino, Tatsuya (1987). A Typed Lambda Calculus with Categorical Type Constructors. In: *Category Theory and Computer Science*. Edited by David H. Pitt, Axel Poigné, and David E. Rydeheard. Vol. 283. Lecture Notes in Computer Science. Springer, pp. 140–157. DOI: 10.1007/3-540-18508-9_24. Cited on pp. 62, 63.

Hughes, John and Andrew Moran (1995). Making Choices Lazily. In: *Proceedings of the 7th International Conference on Functional Programming Languages and Computer Architecture, FPCA 1995*. Edited by John Williams. ACM Press, pp. 108–119. DOI: 10.1145/224164.224191. Cited on p. 62.

Hur, Chung-Kil, Georg Neis, Derek Dreyer, and Viktor Vafeiadis (2013). The power of parameterization in coinductive proof. In: *The 40th Annual ACM Symposium on Principles of Programming Languages, POPL'13*. Edited by Roberto Giacobazzi and Radhia Cousot. ACM Press, pp. 193–206. DOI: 10.1145/2429069.2429093. Cited on pp. 63, 224, 231.

Igarashi, Atsushi and Hideshi Nagira (2007). Union Types for Object-Oriented Programming. In: *Journal of Object Technology* 6.2, pp. 47–68. DOI: 10.5381/jot.2007.6.2.a3. Cited on p. 113.

Igarashi, Atsushi, Benjamin C. Pierce, and Philip Wadler (2001). Featherweight Java: A Minimal Core Calculus for Java and GJ. In: *ACM Transactions on Programming Languages and Systems* 23.3, pp. 396–450. DOI: 10.1145/503502.503505. Cited on pp. 113, 223.

Jeannin, Jean-Baptiste and Dexter Kozen (2012). Computing with Capsules. In: *Journal of Automata, Languages and Combinatorics* 17.2-4, pp. 185–204. DOI: 10.25596/jalc-2012-185. Cited on p. 224.

Jeannin, Jean-Baptiste, Dexter Kozen, and Alexandra Silva (2013). Language Constructs for Non-Well-Founded Computation. In: *Programming Languages and Systems - 22nd European Symposium on Programming, ESOP 2013*. Edited by Matthias Felleisen and Philippa Gardner. Vol. 7792. Lecture Notes in Computer Science. Springer, pp. 61–80. DOI: 10.1007/978-3-642-37036-6_4. Cited on pp. 62, 223.

Jeannin, Jean-Baptiste, Dexter Kozen, and Alexandra Silva (2017). CoCaml: Functional Programming with Regular Coinductive Types. In: *Fundamenta Informaticae* 150.3-4, pp. 347–377. DOI: 10.3233/FI-2017-1473. Cited on pp. 62, 223, 224.

Johann, Patricia, Ekaterina Komendantskaya, and Vladimir Komendantskiy (2015). Structural Resolution for Logic Programming. In: *Proceedings of the Technical Communications of the 31st International Conference on Logic Programming, ICLP 2015*. Edited by Marina De Vos, Thomas Eiter, Yuliya Lierler, and Francesca Toni. Vol. 1433. CEUR Workshop Proceedings. CEUR-WS.org. Cited on pp. 62, 223.

Kahn, Gilles (1987). Natural Semantics. In: *4th Annual Symposium on Theoretical Aspects of Computer Science, STACS'87*. Edited by Franz-Josef Brandenburg, Guy Vidal-Naquet, and Martin Wirsing. Vol. 247. Lecture Notes in Computer Science. Springer, pp. 22–39. DOI: 10.1007/BFb0039592. Cited on p. 69.

Komendantskaya, Ekaterina, Patricia Johann, and Martin Schmidt (2016). A Productivity Checker for Logic Programming. In: *Logic-Based Program Synthesis and Transformation - 26th International Symposium, LOPSTR 2016*. Edited by Manuel V. Hermenegildo and Pedro López-García. Vol. 10184. Lecture

Notes in Computer Science. Springer, pp. 168–186. DOI: 10.1007/978-3-319-63139-4_10. Cited on p. 223.

Komendantskaya, Ekaterina and Yue Li (2017). Productive corecursion in logic programming. In: *Theory and Practice of Logic Programming* 17.5-6, pp. 906–923. DOI: 10.1017/S147106841700028X. Cited on p. 223.

Komendantskaya, Ekaterina, John Power, and Martin Schmidt (2016). Coalgebraic logic programming: from Semantics to Implementation. In: *Journal of Logic and Computation* 26.2, pp. 745–783. DOI: 10.1093/logcom/exu026. Cited on p. 223.

Leroy, Xavier and Hervé Grall (2009). Coinductive big-step operational semantics. In: *Information and Computation* 207.2, pp. 284–304. DOI: 10.1016/j.ic.2007.12.004. Cited on pp. 11, 62, 83, 121, 167, 168, 170.

Leroy, Xavier and François Rouaix (1998). Security Properties of Typed Applets. In: *The 25th Annual ACM Symposium on Principles of Programming Languages, POPL'98*. Edited by David B. MacQueen and Luca Cardelli. ACM Press, pp. 391–403. DOI: 10.1145/268946.268979. Cited on p. 62.

Li, Yue (2017). Structural Resolution with Co-inductive Loop Detection. In: *Proceedings of the First Workshop on Coalgebra, Horn Clause Logic Programming and Types, CoALP-Ty'16*. Edited by Ekaterina Komendantskaya and John Power. Vol. 258. Electronic Proceedings in Theoretical Computer Science, pp. 52–67. DOI: 10.4204/EPTCS.258.4. Cited on p. 223.

Lloyd, John W. (1987). *Foundations of Logic Programming, 2nd Edition*. Springer. DOI: 10.1007/978-3-642-83189-8. Cited on pp. 204, 205, 208.

Löding, Christof and Andreas Tollkötter (2016). Transformation Between Regular Expressions and omega-Automata. In: *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016*. Edited by Piotr Faliszewski, Anca Muscholl, and Rolf Niedermeier. Vol. 58. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 88:1–88:13. DOI: 10.4230/LIPIcs.MFCS.2016.88. Cited on p. 210.

Mac Lane, Saunders (1978). *Categories for the Working Mathematician*. Graduate Texts in Mathematics 5. Springer. DOI: 10.1007/978-1-4757-4721-8. Cited on p. 134.

Mantadelis, Theofrastos, Ricardo Rocha, and Paulo Moura (2014). Tabling, Rational Terms, and Coinduction Finally Together! In: *Theory and Practice of Logic Programming* 14.4-5, pp. 429–443. DOI: 10.1017/S147106841400012X. Cited on pp. 62, 223.

Marino, Daniel and Todd D. Millstein (2009). A generic type-and-effect system. In: *Proceedings of TLDI'09: ACM International Workshop on Types in Languages Design and Implementation*. Edited by Andrew Kennedy and Amal Ahmed. ACM Press, pp. 39–50. DOI: 10.1145/1481861.1481868. Cited on pp. 171, 173.

Markowsky, George (1976). Chain-complete posets and directed sets with applications. In: *Algebra Universalis* 6.1, pp. 53–68. Cited on p. 129.

McBride, Conor (2015). Turing-Completeness Totally Free. In: *Mathematics of Program Construction - 12th International Conference, MPC 2015*. Edited

by Ralf Hinze and Janis Voigtländer. Vol. 9129. Lecture Notes in Computer Science. Springer, pp. 257–275. DOI: 10.1007/978-3-319-19797-5_13. Cited on p. 171.

Milius, Stefan, Dirk Pattinson, and Thorsten Wißmann (2016). A New Foundation for Finitary Corecursion - The Locally Finite Fixpoint and Its Properties. In: *Foundations of Software Science and Computation Structures - 19th International Conference, FOSSACS 2016*. Edited by Bart Jacobs and Christof Löding. Vol. 9634. Lecture Notes in Computer Science. Springer, pp. 107–125. DOI: 10.1007/978-3-662-49630-5_7. Cited on p. 180.

Milius, Stefan, Dirk Pattinson, and Thorsten Wißmann (2019). A New Foundation for Finitary Corecursion and Iterative Algebras. In: *Information and Computation*, p. 104456. DOI: 10.1016/j.ic.2019.104456. Cited on p. 180.

Milner, Robin (1978). A Theory of Type Polymorphism in Programming. In: *Journal of Computer and System Sciences* 17.3, pp. 348–375. DOI: 10.1016/0022-0000(78)90014-4. Cited on p. 95.

Milner, Robin and Mads Tofte (1991). Co-Induction in Relational Semantics. In: *Theoretical Computer Science* 87.1, pp. 209–220. DOI: 10.1016/0304-3975(91)90033-X. Cited on p. 62.

Moerdijk, Ieke and Erik Palmgren (2000). Wellfounded trees in categories. In: *Annals of Pure and Applied Logic* 104.1-3, pp. 189–218. DOI: 10.1016/S0168-0072(00)00012-9. Cited on pp. 13, 14.

Momigliano, Alberto and Alwen Fernanto Tiu (2003). Induction and Coinduction in Sequent Calculus. In: *Types for Proofs and Programs, International Workshop, TYPES 2003*. Edited by Stefano Berardi, Mario Coppo, and Ferruccio Damiani. Vol. 3085. Lecture Notes in Computer Science. Springer, pp. 293–308. DOI: 10.1007/978-3-540-24849-1_19. Cited on pp. 63, 64.

Moura, Paulo (2013). A Portable and Efficient Implementation of Coinductive Logic Programming. In: *Practical Aspects of Declarative Languages - 15th International Symposium, PADL 2013*. Edited by Konstantinos Sagonas. Vol. 7752. Lecture Notes in Computer Science. Springer, pp. 77–92. DOI: 10.1007/978-3-642-45284-0_6. Cited on p. 223.

Nakata, Keiko and Tarmo Uustalu (2009). Trace-Based Coinductive Operational Semantics for While. In: *Theorem Proving in Higher Order Logics, 22nd International Conference, TPHOLs 2009*. Edited by Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel. Vol. 5674. Lecture Notes in Computer Science. Springer, pp. 375–390. DOI: 10.1007/978-3-642-03359-9_26. Cited on p. 170.

Nakata, Keiko and Tarmo Uustalu (2010a). A Hoare Logic for the Coinductive Trace-Based Big-Step Semantics of While. In: *Programming Languages and Systems - 19th European Symposium on Programming, ESOP 2010*. Edited by Andrew D. Gordon. Vol. 6012. Lecture Notes in Computer Science. Springer, pp. 488–506. DOI: 10.1007/978-3-642-11957-6_26. Cited on p. 170.

Nakata, Keiko and Tarmo Uustalu (2010b). Resumptions, Weak Bisimilarity and Big-Step Semantics for While with Interactive I/O: An Exercise in Mixed Induction-Coinduction. In: *Proceedings 7th Workshop on Structural Opera-*

tional Semantics, SOS 2010. Edited by Luca Aceto and Pawel Sobocinski. Vol. 32. Electronic Proceedings in Theoretical Computer Science, pp. 57–75. DOI: 10.4204/EPTCS.32.5. Cited on p. 170.

Owens, Scott, Magnus O. Myreen, Ramana Kumar, and Yong Kiam Tan (2016). Functional Big-Step Semantics. In: *Programming Languages and Systems - 25th European Symposium on Programming, ESOP 2016*. Edited by Peter Thiemann. Vol. 9632. Lecture Notes in Computer Science. Springer, pp. 589–615. DOI: 10.1007/978-3-662-49498-1_23. Cited on pp. 169, 170, 172.

Perrin, Dominique and Jean-Éric Pin (2004). *Infinite Words: Automata, Semigroups, Logic and Games*. Pure and Applied Mathematics Bd. 141. Elsevier. Cited on pp. 122, 126, 171.

Pierce, Benjamin C. (2002). *Types and programming languages*. MIT Press. Cited on pp. 70, 83, 86, 168.

Piróg, Maciej and Jeremy Gibbons (2014). The Coinductive Resumption Monad. In: *Proceedings of the 30th Conference on the Mathematical Foundations of Programming Semantics, MFPS 2014*. Vol. 308. Electronic Notes in Theoretical Computer Science. Elsevier, pp. 273–288. DOI: 10.1016/j.entcs.2014.10.015. Cited on p. 170.

Plotkin, Gordon D. (1981). *A structural approach to operational semantics*. Technical Report. Aarhus University. Cited on p. 69.

Plotkin, Gordon D. (2004). A Structural Approach to Operational Semantics. In: *Journal of Logic and Algebraic Programming* 60-61, pp. 17–139. Cited on p. 69.

Poulsen, Casper Bach and Peter D. Mosses (2017). Flag-based Big-step Semantics. In: *Journal of Logic and Algebraic Methods in Programming* 88, pp. 174–190. DOI: 10.1016/j.jlamp.2016.05.001. Cited on pp. 169, 172.

Pous, Damien (2007). Complete Lattices and Up-To Techniques. In: *Programming Languages and Systems, 5th Asian Symposium, APLAS 2007*. Edited by Zhong Shao. Vol. 4807. Lecture Notes in Computer Science. Springer, pp. 351–366. DOI: 10.1007/978-3-540-76637-7_24. Cited on pp. 63, 224, 231.

Pous, Damien (2016). Coinduction All the Way Up. In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2016*. Edited by Martin Grohe, Eric Koskinen, and Natarajan Shankar. ACM Press, pp. 307–316. DOI: 10.1145/2933575.2934564. Cited on pp. 63, 224, 231.

Pous, Damien and Davide Sangiorgi (2012). Enhancements of the bisimulation proof method. In: *Advanced Topics in Bisimulation and Coinduction*. Edited by Davide Sangiorgi and Jan Rutten. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, pp. 233–289. DOI: 10.1017/CBO9780511792588.007. Cited on pp. 63, 224, 231.

Reynolds, John C. (1972). Definitional interpreters for higher-order programming languages. In: *ACM'72, Proceedings of the ACM annual conference*. Vol. 2. ACM Press, pp. 717–740. Cited on p. 169.

Rutten, Jan J. M. M. (2000). Universal coalgebra: a theory of systems. In: *Theoretical Computer Science* 249.1, pp. 3–80. DOI: 10.1016/S0304-3975(00)00056-6. Cited on p. 86.

Sangiorgi, Davide (2011). *Introduction to Bisimulation and Coinduction*. Cambridge University Press. Cited on pp. 11, 12, 21, 62.

Santocanale, Luigi (2002). A Calculus of Circular Proofs and Its Categorical Semantics. In: *Foundations of Software Science and Computation Structures - 5th International Conference, FOSSACS 2002*. Edited by Mogens Nielsen and Uffe Engberg. Vol. 2303. Lecture Notes in Computer Science. Springer, pp. 357–371. DOI: 10.1007/3-540-45931-6_25. Cited on p. 222.

Simon, Luke (2006). *Extending logic programming with coinduction*. PhD thesis. University of Texas at Dallas. Cited on pp. 203–208, 221, 223.

Simon, Luke, Ajay Mallya, Ajay Bansal, and Gopal Gupta (2006). Coinductive Logic Programming. In: *Logic Programming, 22nd International Conference, ICLP 2006*. Edited by Sandro Etalle and Miroslaw Truszczynski. Vol. 4079. Lecture Notes in Computer Science. Springer, pp. 330–345. DOI: 10.1007/11799573_25. Cited on pp. 57, 62, 178, 204–208, 221, 223.

Simon, Luke, Ajay Bansal, Ajay Mallya, and Gopal Gupta (2007). Co-Logic Programming: Extending Logic Programming with Coinduction. In: *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007*. Edited by Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki. Vol. 4596. Lecture Notes in Computer Science. Springer, pp. 472–483. DOI: 10.1007/978-3-540-73420-8_42. Cited on pp. 56, 57, 62, 203–205, 207, 208, 221, 223.

Spadotti, Régis (2016). *A mechanized theory of regular trees in dependent type theory. (Une théorie mécanisée des arbres réguliers en théorie des types dépendants)*. PhD thesis. Paul Sabatier University, Toulouse, France. Cited on p. 224.

Tarski, Alfred (1955). A lattice-theoretical fixpoint theorem and its applications. In: *Pacific Journal of Mathematics* 5.2, pp. 285–309. Cited on p. 20.

Tate, Ross (2013). The sequential semantics of producer effect systems. In: *The 40th Annual ACM Symposium on Principles of Programming Languages, POPL'13*. Edited by Roberto Giacobazzi and Radhia Cousot. ACM Press, pp. 15–26. DOI: 10.1145/2429069.2429074. Cited on pp. 171, 173.

The Agda Team. *The Agda Reference Manual*. Cited on p. 63.

The Coq Development Team. *The Coq Reference Manual*. INRIA. Cited on p. 63.

Uustalu, Tarmo and Niccolò Veltri (2017). Finiteness and rational sequences, constructively. In: *Journal of Functional Programming* 27, e13. DOI: 10.1017/S0956796817000041. Cited on p. 224.

van den Berg, Benno and Federico De Marchi (2007). Non-well-founded trees in categories. In: *Annals of Pure and Applied Logic* 146.1, pp. 40–59. DOI: 10.1016/j.apal.2006.12.001. Cited on pp. 13, 14.

Wright, Andrew K. and Matthias Felleisen (1994). A Syntactic Approach to Type Soundness. In: *Information and Computation* 115.1, pp. 38–94. DOI: 10.1006/inco.1994.1093. Cited on pp. 69, 95, 97.

Xia, Li-yao, Yannick Zakowski, Paul He, Chung-Kil Hur, Gregory Malecha, Benjamin C. Pierce, and Steve Zdancewic (2020). Interaction trees: representing recursive and impure programs in Coq. In: *Proceedings of ACM on Programming Languages* 4.The 47th Annual ACM Symposium on Principles

of Programming Languages, POPL'20, 51:1–51:32. DOI: 10.1145/3371119. Cited on p. 171.