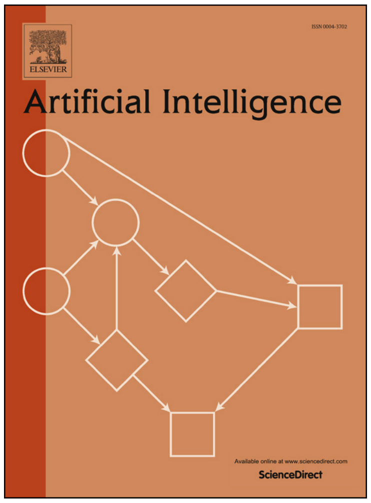
# A Simple Proof-theoretic Characterization of Stable Models: Reduction to Difference Logic and Experiments

Martin Gebser, Enrico Giunchiglia, Marco Maratea and Marco Mochi



PII: S0004-3702(24)00212-1
DOI: https://doi.org/10.1016/j.artint.2024.104276
Reference: ARTINT 104276

To appear in: *Artificial Intelligence*

Received date: 21 January 2023
Revised date: 14 December 2024
Accepted date: 16 December 2024

# A Simple Proof-theoretic Characterization of Stable Models: Reduction to Difference Logic and Experiments

Martin Gebser[a], Enrico Giunchiglia[b], Marco Maratea[c], Marco Mochi[b]

[a]*AICS, Universität Klagenfurt, Klagenfurt, Austria,*
[b]*DIBRIS, University of Genoa, Italy,*
[c]*DeMaCS, University of Calabria, Rende, Italy,*

**Abstract**

Stable models of logic programs have been studied and characterized in relation with other formalisms by many researchers. As already argued in previous papers, such characterizations are interesting for diverse reasons, including theoretical investigations and the possibility of leading to new algorithms for computing stable models of logic programs. At the theoretical level, complexity and expressiveness comparisons have brought about fundamental insights. Beyond that, practical implementations of the developed reductions enable the use of existing solvers for other logical formalisms to compute stable models. In this paper, we first provide a simple characterization of stable models that can be viewed as a proof-theoretic counterpart of the standard model-theoretic definition. We further show how it can be naturally encoded in difference logic. Such an encoding, compared to the existing reductions to classical logics, does not require Boolean variables. Then, we implement our novel translation to a Satisfiability Modulo Theories (SMT) formula. We finally compare our approach, employing the SMT solver YICES, to the translation-based ASP solver LP2DIFF and to CLINGO on domains from the "Basic Decision" track of the 2017 Answer Set Programming competition. The results show that our approach is competitive to and often better than LP2DIFF, and that it can also be faster than CLINGO on non-tight domains.

*Keywords:* Logic programming, stable models, answer set programming, difference logic

## 1. Introduction

Logic programming and non-monotonic reasoning are two important and historical areas within Artificial Intelligence, which have been studied and are at the core of knowledge representation and reasoning. During the years, many theoretical contributions have been presented; relatively more recently, solving tools, in particular for Answer Set Programming (ASP) (Baral, 2003; Brewka et al., 2011; Gelfond and Lifschitz, 1988, 1991; Marek et al., 2008; Niemelä, 1999), have been employed to solve real-life problems, even in industrial settings (Erdem et al., 2016; Falkner et al., 2018; Schüller, 2018; Gebser et al., 2018; Dodaro et al., 2021). Thus, it is often important to

*Email addresses:* `martin.gebser@aau.at` (Martin Gebser), `enrico.giunchiglia@unige.it` (Enrico Giunchiglia), `marco.maratea@unige.it` (Marco Maratea), `marco.mochi@edu.unige.it` (Marco Mochi)

complement theoretical findings with the implementation and analysis of related solving systems (see, e.g., (Gebser et al., 2012; Alviano et al., 2019; Janhunen et al., 2009) for ASP solvers). Stable models of logic programs, a.k.a. answer sets, have been studied and characterized also in relation with other formalisms. As already argued (see, e.g., (Lifschitz, 2010)), such characterizations are interesting for diverse reasons, including the possibility of computing stable models by means of reductions enabling the use of existing solvers for other logical formalisms. This is the case, e.g., for the SAT-based approaches of the ASSAT (Lin and Zhao, 2004) and CMODELS (Giunchiglia et al., 2006) ASP solvers, and the reduction from logic programs to Mixed Integer Programming implemented by LP2MIP (Liu et al., 2012).

In this paper, we first introduce stable derivations as a new characterization of stable models and show how it can be naturally encoded in difference logic, i.e., as quantifier-free first-order formulas whose atoms have the form

$$(x \bowtie y + c),$$

where $x$ and $y$ are variables ranging over the reals/rationals or the integers, $c$ is a numeric constant, and $\bowtie \in \{=, \neq, \leq, <, \geq, >\}$. While this is neither the first alternative to the standard definition of stable models (see, e.g., (Lifschitz, 2010)) nor the first reduction to difference logic (see, e.g., (Niemelä, 2008)),

1. our definition of stable derivation is simple, as witnessed by the fact that it is rather short, and
2. its corresponding reduction to difference logic uses only one numeric variable per atom in a logic program, without the need to include any Boolean variable.

Stable derivations can be viewed as a proof-theoretic counterpart of the standard model-theoretic definition of stable models. About the reduction, we first provide a basic version directly following the characterization. Then, we introduce an improved reduction that takes Strongly Connected Components (SCCs) into account, which come into play for distinguishing recursive positive body atoms on non-tight domains (Erdem and Lifschitz, 2003).

We implement these translations to Satisfiability Modulo Theories (SMT) formulas[1] expressed in difference logic. Employing the SMT solver YICES (Dutertre, 2014a), we compare our reduction to the translation-based ASP solver LP2DIFF, which comes certainly closest to our approach, and to the state-of-the-art solver CLINGO on domains from the 2017 ASP competition (Gebser et al., 2020). In particular, we consider all domains of the "Basic Decision" track of the competition, whose ASP encodings consist of normal rules with classical and built-in atoms only, i.e., the type of logic programs we deal with in our paper. The results show that, when employing YICES, our approach is competitive to and often better than LP2DIFF, and that it can be also faster than CLINGO on non-tight domains. However, let us also point out that, beyond the direct use of our reduction to compute stable models by SMT solvers, one may take it as basis for future extensions by theory reasoning (Lierler, 2023; Kaminski et al., 2023), e.g., on floating point numbers or even real arithmetic.

[1] http://smtlib.cs.uiowa.edu/standard.shtml

The paper is structured as follows. First, Section 2 introduces necessary preliminaries. Section 3 then presents our characterization of stable derivations, while the corresponding reduction to difference logic is provided in Section 4. In Section 5, we develop an improved version of this reduction by taking SCCs into account. Implementation details and results of our experiments are described in Section 6. Section 7 and 8 conclude the paper by discussing further related work, and by drawing some conclusions and possible topics for future research, respectively.

## 2. Stable models, Clark's completion and ordering constraints

Let $V$ be a countable set of *atoms*. By $V^\perp$ we mean the set obtained adding the atom $\perp$ denoting falsity to $V$, i.e., $V^\perp = V \cup \{\perp\}$. A *rule* is an expression of the form

$$A \leftarrow A_1, \dots, A_m, \neg A_{m+1}, \dots, \neg A_n \tag{1}$$

($0 \leq m \leq n$) where $A, A_1, \dots, A_n$ are atoms in $V^\perp$ and $\neg$ is the symbol for negation. We assume that in (1), $A_i \neq A_j$ for $1 \leq i < j \leq n$. A *(logic) program* is a set of rules. Given a rule $r$ of the form (1), $\mathit{head}(r) = A$ is the *head*, $\mathit{body}^+(r) = \{A_1, \dots, A_m\}$ is the set of positive body atoms, $\mathit{body}^-(r) = \{A_{m+1}, \dots, A_n\}$ is the set of negative body atoms, and $\mathit{body}(r) = \{A_1, \dots, A_m, \neg A_{m+1}, \dots, \neg A_n\}$ is the *body* of $r$. If $A = \perp$ then (1) is said to be a *constraint*.

Consider a logic program $\Pi$. A *(truth) assignment* is a subset of the set $V$ of atoms, thus not containing $\perp$. A truth assignment $M$ *satisfies*

1. an atom $A$ if $A \in M$,
2. a negated atom $\neg A$ if $A \notin M$,
3. a set of atoms and negated atoms if $M$ satisfies all the elements in the set,
4. a rule if $M$ satisfies the head whenever $M$ satisfies the body of the rule,
5. a program $\Pi$ if $M$ satisfies all the rules in $\Pi$, in which case $M$ is also said to be a *model* of $\Pi$.

A model $M$ of $\Pi$ is *minimal* if $\Pi$ has no other model which is a subset of $M$. As standard, for a suitable concept $S$, we write $M \models S$ to mean that $M$ satisfies $S$.

For any truth assignment $M$, the *reduct* $\Pi^M$ of $\Pi$ relative to $M$ is the set of rules obtained from $\Pi$ by considering each rule $r \in \Pi$ of the form (1) and dropping $r$ if at least one of the atoms in the negative part of the $\mathit{body}^-(r)$ is in $M$, and then dropping $\neg A_{m+1} \dots, \neg A_n$ otherwise, i.e.,

$$\Pi^M = \{\mathit{head}(r) \leftarrow \mathit{body}^+(r) : r \in \Pi, M \cap \mathit{body}^-(r) = \emptyset\}.$$

A truth assignment $M$ is a *stable model* of $\Pi$ if it is the least model of the reduct of $\Pi$ relative to $M$.

**Example 1.** *Consider the program $\Pi$ in the three atoms $A, B, C$ whose rules are:*

$$\begin{array}{c} A \leftarrow B, \\ B \leftarrow A, \\ A \leftarrow \neg C, \\ C \leftarrow C. \end{array} \tag{2}$$

*$\Pi$ has the three models $\{A, B, C\}$, $\{A, B\}$ and $\{C\}$, of which only the last two are minimal and only the second one is stable.*

A model $M$ is a *supported model* of $\Pi$ if for each atom $A \in V^{\perp}$, $A \in M$ iff there exists a rule $r \in \Pi$ such that $head(r) = A$ and $M \models body(r)$. The provided definition of stable model is due to Gelfond and Lifschitz (1988) and has the property that each stable model $M$ of $\Pi$ is a supported model of $\Pi$. Thus, all the stable models are also supported while the converse is not necessarily true (Marek and Subrahmanian, 1992). Fages (1994) proved that also the converse is true if $\Pi$ is *tight*, i.e., if the *positive dependency graph* of $\Pi$

1. having one node for each atom in $V$, and
2. an edge from $A \neq \perp$ to each atom $A_1 \neq \perp, \ldots, A_m \neq \perp$ for each rule (1) in $\Pi$,

does not contain any loop.

**Theorem 1** (Fages (1994)). *Let $\Pi$ be a program. A stable model of $\Pi$ is also a supported model of $\Pi$, and if $\Pi$ is tight then a supported model of $\Pi$ is also a stable model of $\Pi$.*

If for each atom $A$ there are finitely many rules with head $A$, the supported models of $\Pi$ coincide with the models of the Clark's completion $Comp(\Pi)$. Assuming $\Pi$ is finite, the *Clark's completion* $Comp(\Pi)$ of $\Pi$ is defined to be the set of formulas in propositional logic consisting of

$$A \equiv \bigvee_{r: r \in \Pi, head(r) = A} \bigwedge_{L \in body(r)} L, \tag{3}$$

for each atom $A \in V^{\perp}$.

Note that (3) is included in $Comp(\Pi)$ for every $A \in V^{\perp}$, even when $A = \perp$ or $A$ is not the head of any rule in $\Pi$. In the former case, (3) is equivalent to

$$\neg \bigvee_{r: r \in \Pi, head(r) = \perp} \bigwedge_{L \in body(r)} L,$$

and in the latter case, (3) is equivalent to $\neg A$. Clark's completion provides a reduction to classical logic for tight programs. $Comp(\Pi)$ has $|V|$ Boolean variables and size $O(||\Pi||)$, where $||\Pi||$ is the size of $\Pi$.

Babovich et al. (2000) and later Erdem and Lifschitz (2003) generalized Fages' result to programs $\Pi$ *tight on a set $M \subseteq V$ of atoms*, defined as the programs for which there exists a function $\lambda$ mapping each atom in $M$ to an ordinal such that for each rule $r$ in $\Pi$, if $M$ satisfies the head and the body of the rule then, for each atom $A$ in the positive body of the rule, $\lambda(head(r)) > \lambda(A)$. A program is tight according to Fages' definition, if it is tight on every set of atoms.

**Theorem 2** (Erdem and Lifschitz (2003)). *Let $\Pi$ be a program. Let $M$ be a supported model of $\Pi$. If $\Pi$ is tight on $M$ then $M$ is a stable model of $\Pi$.*

**Example 2.** *Consider the rules in (2). If $\Pi$ consists of the second and third rules then $\Pi$ is tight and $Comp(\Pi)$ consists of the formulas*

$$\begin{gathered} A \equiv \neg C, \\ B \equiv A, \\ \neg C. \end{gathered}$$

*If $\Pi$ consists of the last three rules in (2), then (i) $\Pi$ is not tight, (ii) $Comp(\Pi)$ consists of the first two of the above formulas, (iii) we can conclude that $M = \{A, B\}$ is a stable model since $\Pi$ is tight on $M$, but (iv) we are not allowed to conclude, on the basis of Theorem 2, that $\{C\}$ is not a stable model.*

*If $\Pi$ consists of all the rules in (2), then (i) $\Pi$ is not tight, (ii) $Comp(\Pi)$ consists of the formulas*

$$\begin{array}{r} A \equiv (B \vee \neg C), \\ B \equiv A, \\ C \equiv C, \end{array} \tag{4}$$

*(iii) the set of models of $Comp(\Pi)$ is $\{\{A, B, C\}, \{A, B\}, \{C\}\}$, and (iv) $\Pi$ is not tight on any of the models of $Comp(\Pi)$.*

If the program $\Pi$ is non tight, several authors showed how it possible to add extra constraints in order to rule out the supported models which are not stable, see for instance (Ben-Eliyahu and Dechter, 1994; Lin and Zhao, 2003, 2004).

Here, in the following, we give a brief overview of the approaches which are more related to our work. Other related works are discussed in Section 8.

Janhunen showed that, in order to rule out the models of the completion which are not stable, it is sufficient to add suitable level ordering constraints. For any truth assignment $M \subseteq V$, define the set of *supporting rules of* $M$ to be $\Pi_M = \{r \in \Pi : M \models body(r)\}$. Given an assignment $M$, a *level numbering* of $M$ for $\Pi$ is a function $\lambda : M \cup \Pi_M \mapsto \mathbb{N}$ such that for each atom $A \in M$,

$$\lambda(A) = min\{\lambda(r) : r \in \Pi_M, head(r) = A\}$$

and, for each rule $r \in \Pi_M$,

$$\lambda(r) = max\{0, max\{\lambda(A) : A \in body^+(r)\}\} + 1.$$

**Theorem 3** (Janhunen (2004)). *Let $\Pi$ be a program. Let $M$ be a supported model of $\Pi$. $M$ is a stable model of $\Pi$ iff there exists a level numbering of $M$ for $\Pi$.*

In the same work, Janhunen proved that, for a supported model $M$ of $\Pi$, there is at most one level numbering, and also showed –assuming $V$ is finite– how to encode level numbering in propositional logic using $\lceil log_2(|V|+2)\rceil$ bits. Thanks to Theorem 3, there exists a one-one-correspondence between the stable models of $\Pi$ and the models of $J(\Pi)$, which consists of the encoding of the level numbering and of $Comp(\Pi)$. $J(\Pi)$ has $O(|V| \times \lceil log_2(|V|)\rceil)$ Boolean variables and size $O(||\Pi|| \times log_2(|V|))$.

A few years later, Niemelä introduced *level ranking* of an assignment $M$ for $\Pi$ to be a function $\lambda : M \mapsto \mathbb{N}$ such that for each atom $A \in M$, there exists a rule $r \in \Pi_M$ such that $head(r) = A$ and for each atom $B \in body^+(r)$, $\lambda(A) \geq \lambda(B) + 1$. Niemelä also showed that if we add the restrictions to level rankings saying that for each $A \in M$

1. $\lambda(A) = 1$ whenever there is a rule $r \in \Pi_M$ with $head(r) = A$ and $body^+(r) = \emptyset$, and
2. for every rule $r \in \Pi_M$ with $head(r) = A$ and $body^+(r) \neq \emptyset$, there exists $B \in body^+(r)$ with $\lambda(A) \leq \lambda(B) + 1$,

then we have a one-to-one correspondence between level ranking and level numbering. Level rankings satisfying such additional restrictions are said to be *strong*.

**Theorem 4** (Niemelä (2008)). *Let $\Pi$ be a program. Let $M$ be a supported model of $\Pi$. $M$ is a stable model of $\Pi$ iff there exists a (strong) level ranking of $M$ for $\Pi$.*

Like level numbering, for each supported model $M$, there is at most one strong level ranking. The strong level ranking can be thus used to produce compact encodings in propositional logic as in Janhunen (2004), but without the need of encoding the level associated to the rules.

(Strong) level rankings can be encoded in difference logic, defined as the extension to propositional logic in which the set of atomic formulas is extended in order to allow for expressions of the form $x \bowtie y+c$, where $x$ and $y$ are variables ranging over a numeric unbounded domain (usually the integers or the rationals/reals), $c$ is a numeric constant and $\bowtie \in \{=, \neq, \leq, <, \geq, >\}$. Then, an *interpretation* $\sigma$ maps each numeric variable to a value in its domain, and $\sigma$ *satisfies* an atomic formula $x \bowtie y + c$ iff $\sigma(x) \bowtie \sigma(y) + c$.[2] Thanks to Theorem 4, the stable models of $\Pi$ can be computed as the models of $N(\Pi)$, where $N(\Pi)$ is the set of formulas in difference logic consisting of $Comp(\Pi)$ and of the encoding of the (strong) level ranking. $N(\Pi)$ has $|V|$ Boolean variables, $|V|$ numeric variables and size $O(||\Pi||)$, though the introduction of additional Boolean variables may produce a more compact encoding, but still in $O(||\Pi||)$.

**Example 3.** *Let $\Pi$ be the set of rules in (2). For each atom $A \in V$, we assume to have a numeric variable $\lambda_N(A)$ in the difference logic encoding. Then, $N(\Pi)$, as defined in Niemelä (2008), is equivalent to*

$$\begin{array}{c} A \equiv (B \vee \neg C), \\ B \equiv A, \\ C \equiv C, \\ A \rightarrow ((B \wedge \lambda_N(A) \geq \lambda_N(B) + 1) \vee \neg C), \\ B \rightarrow A \wedge \lambda_N(B) \geq \lambda_N(A) + 1, \\ C \rightarrow C \wedge \lambda_N(C) \geq \lambda_N(C) + 1, \end{array} \tag{5}$$

*where the first 3 formulas correspond to $Comp(\Pi)$ and the other ones are the encoding of the level ranking conditions. Any model of the above formulas satisfy $A, B, \neg C, \lambda_N(B) \geq \lambda_N(A) + 1$.*

*The formulas encoding the additional conditions on strong level ranking are*

$$\begin{array}{c} A \rightarrow (\neg B \vee \lambda_N(A) \leq \lambda_N(B) + 1) \wedge (C \vee \lambda_N(A) = \lambda_N(\top)), \\ B \rightarrow \neg A \vee \lambda_N(B) \leq \lambda_N(A) + 1, \\ C \rightarrow \lambda_N(C) \leq \lambda_N(C) + 1, \end{array} \tag{6}$$

[2] It is possible to distinguish between *rational/real difference logic* and *integer difference logic*; in the former, variables take values in the rationals/reals while in the latter case variables are assumed to take integer values. The distinction is useful as, e.g., the satisfiability of $0 < x - y < 1$ depends on the domain of $x$ and $y$. However, in this paper such distinction is useless since we are going to consider formulas whose satisfiability does not depend on the chosen domain.

*where $\lambda_N(\top)$ is a "dummy" variable necessary in order to respect the syntax of difference logic and whose intended interpretation is 1. The encoding in difference logic of the strong level ranking corresponds to the formulas in (5) and (6) which impose, assuming the intended interpretation of $\lambda_N(\top)$, that the models satisfy*

$$\lambda_N(A) = 1,$$
$$\lambda_N(B) = 2.$$

*As expected, strong level rankings (like level numbering) are unique: each atom in the stable model has a uniquely associated level, while the constraints say nothing about the value of the variables associated to the atoms not belonging to the stable model, in this case $\lambda_N(C)$.*

Several other characterizations and corresponding reductions can be introduced on the basis of Niemelä's level ranking. For instance, in the same paper Niemelä defines other reductions based on the SCCs of the positive dependency graph associated to $\Pi$, and Gebser et al. (2014) show how it is possible to encode (strong) level rankings (also exploiting SCCs) in SAT modulo acyclicity. We will also show, in Section 5, how our characterization and enconding, presented in the next two sections, can be improved by handling SCCs.

## 3. A simple proof-theoretic characterization of stable models

This section presents our characterization of stable models, starting from the definition of *stable derivation*, which conceptually relates to Clark's completion (Clark, 1978).

**Definition 1.** *Consider a program $\Pi$. A stable derivation is a function $\lambda$ mapping each atom $A \in V^\perp$ to an ordinal such that $\lambda(A) < \lambda(\perp)$ iff there exists a rule $r \in \Pi$ with head $A$ and*

1. *for each atom $B \in body^+(r)$, $\lambda(A) > \lambda(B)$, and*
2. *for each atom $B \in body^-(r)$, $\lambda(B) \geq \lambda(\perp)$.*

Given a stable derivation $\lambda$, the set of atoms *stably derived by* $\lambda$ is $\{A : \lambda(A) < \lambda(\perp)\}$. A set of atoms $M$ is *stably derivable (from* $\Pi$) if there exists a stable derivation of $M$. From the above definitions, it immediately follows that, in a stable derivation, $\lambda(\perp) = 0$ only if the set of stably derivable atoms is empty.

**Example 4.** *In the case of the logic program (2), every stable derivation $\lambda$ is such that $\lambda(A) < \lambda(B) < \lambda(\perp) \leq \lambda(C)$ and the only stably derivable set of atoms is $\{A, B\}$. In general, there is more than one stably derivable set of atoms, as in the case of the program*

$$A \leftarrow \neg B,$$
$$B \leftarrow \neg A$$

*whose stable derivations satisfy either*

$$\lambda(A) < \lambda(\perp) \leq \lambda(B)$$

*or*

$$\lambda(B) < \lambda(\perp) \leq \lambda(A)$$

*and the two corresponding stably derivable sets of atoms are $\{A\}$ and $\{B\}$.*

A set of atoms is stably derivable iff it is a stable model.

**Theorem 5.** *Let* $\Pi$ *be a program. A set of atoms* $M$ *is a stable model of* $\Pi$ *iff* $M$ *is stably derivable from* $\Pi$.

*Proof.* For the left to right direction, assume $M$ is a stable model. We define a stable derivation for $\Pi$ via the operator $T_{\Pi^M} : 2^V \mapsto 2^V$ defined, for an arbitrary program $\Pi$, as

$$T_\Pi(I) = \{head(r) : r \in \Pi, I \models body(r)\},$$

and considering the following sequence of subsets of $V$

$$T_\Pi{\uparrow}^0 = \emptyset,$$

and, for each $i \geq 0$,

$$T_\Pi{\uparrow}^{i+1} = T_\Pi(T_\Pi{\uparrow}^i).$$

Now, since $M$ is a stable model of $\Pi$, for each $A \in M$ there is a unique $i$ such that $A \in T_{\Pi^M}{\uparrow}^i \setminus T_{\Pi^M}{\uparrow}^{i-1}$, and we set $\lambda(A) = i$ iff $A \in T_{\Pi^M}{\uparrow}^i \setminus T_{\Pi^M}{\uparrow}^{i-1}$. For $A \notin M$, we set $\lambda(A) = \lambda(\bot) = \omega$, the first limit ordinal. Then, $M = T_{\Pi^M}{\uparrow}^\omega$ and for each $A \in V$, $\lambda(A) < \lambda(\bot)$ iff $A \in M$. Clearly, $\lambda$ is a stable derivation for $\Pi$: if $\lambda(A) = i < \omega$ then $A \in T_{\Pi^M}{\uparrow}^i \setminus T_{\Pi^M}{\uparrow}^{i-1}$ and thus there exists a rule $r \in \Pi$ such that $(i)$ for each $B \in body^+(r)$, $B \in T_{\Pi^M}{\uparrow}^{i-1}$ and thus $\lambda(B) < \lambda(A)$, and $(ii)$ for each $B \in body^-(r)$, $B \notin M$ and thus $\lambda(B) = \lambda(\bot) = \omega$.

For the right to left direction, suppose there is a stable derivation $\lambda$ for $\Pi$. We show that $M = \{A : \lambda(A) < \lambda(\bot)\}$ is the least model of $\Pi^M$, which implies that $M$ is a stable model of $\Pi$. We first show that $M$ is a model of $\Pi^M$. Assume it is not. Then, there exists a rule $r$ of the form (1) such that $M \models \{A_1, \dots, A_m, \neg A_{m+1}, \dots, \neg A_n, \neg A\}$ i.e., $\lambda(A_1) < \lambda(\bot)$, $\dots$, $\lambda(A_m) < \lambda(\bot)$ while $\lambda(A_{m+1}) \geq \lambda(\bot)$, $\dots$, $\lambda(A_n) \geq \lambda(\bot)$, $\lambda(A) \geq \lambda(\bot)$, which implies $\lambda(A_1) < \lambda(A)$, $\dots$, $\lambda(A_m) < \lambda(A)$ and $\lambda(A_{m+1}) \geq \lambda(\bot)$, $\dots$, $\lambda(A_n) \geq \lambda(\bot)$, $\lambda(A) \geq \lambda(\bot)$, which is not possible since $\lambda$ is a stable derivation. Now we prove that $M$ is minimal. Assume it is not. Then, there is another model $M' \subset M$ of $\Pi^M$ and an atom $A \in M \setminus M'$ with the lowest value $\lambda(A)$ among the atoms in $M \setminus M'$. Since $A \in M$ then $\lambda(A) < \lambda(\bot)$ and there exists a rule $r \in \Pi$ such that $M \models body(r)$. But, for each $B \in body^+(r)$, $B \in M'$ since $\lambda(B) < \lambda(A)$, and for each $B \in body^-(r)$, $B \notin M'$ since $M' \subset M$. Thus, $M' \models body(r)$ and then, since $M'$ is a model of $\Pi$, $A \in M'$, contradicting the assumption. □

The term "stable derivation" has been used given $(i)$ the analogy with the standard definition of derivation in classical logic, and $(ii)$ the correspondence, as established by Theorem 5, with stable models. Indeed, a stable derivation can be seen as a sequence of applications of rules as in a standard derivation in classical logic, once

1. each rule $r$ is interpreted as the inference rule $head(r) \leftarrow body^+(r)$ carrying the restriction that the whole derivation must not contain the atoms in $body^-(r)$, and
2. each applicable rule $r$ is applied in the derivation.

Differently from classical logic, given the restrictions of the rules, the later application of an applicable rule $r$ may invalidate the (stability of the) derivation. These two differences make the stably derivable relation nonmonotonic –while classical logic is indeed monotonic– but thanks to Theorem 5 we have a nice correspondence between the standard "model-theoretic" definition of stable model and this "proof-theoretic" definition of stable derivation, again similarly to what happens in classical logic.

Comparing the statements of Theorem 2, Theorem 3, and Theorem 4 with Theorem 5, our characterization of stable models does not assume that the starting assignment $M$ is a supported model. If instead we consider our definition of stable derivation, since for any two ordinals $\alpha$ and $\beta$ the condition $\alpha > \beta$ is equivalent to $\alpha \geq \beta + 1$, it is easy to check the correspondence between the first condition on stable derivation and the condition on level ranking.

As it happens for level rankings, given the freedom in selecting the ordinal associated to each atom, the number of stable derivations is, in general, infinite even in the case of finite programs. If we consider two stable derivations $\lambda_1$ and $\lambda_2$ to be *equivalent* if, for each pair of atoms $A, B \in V^{\perp}$, $\lambda_1(A) < \lambda_1(B)$ iff $\lambda_2(A) < \lambda_2(B)$, then, whenever $V^{\perp}$ is finite, there are finitely many non equivalent stable derivations. It is however still possible that there exists two non equivalent stable derivations having the same set of stably derivable atoms.

**Example 5.** *Consider the logic program $\Pi$ obtained adding $B \leftarrow \neg C$ to (2). In this case there are three sets of non equivalent stable derivations $\lambda_1$, $\lambda_2$ and $\lambda_3$ for $\Pi$, characterized by $\lambda_1(A) < \lambda_1(B) < \lambda_1(\perp) \leq \lambda_1(C)$, $\lambda_2(B) < \lambda_2(A) < \lambda_2(\perp) \leq \lambda_2(C)$ and $\lambda_3(A) = \lambda(B) < \lambda(\perp) \leq \lambda(C)$. However, all the stable derivations lead to the same set $\{A, B\}$ of stably derivable atoms.*

We can thus define a weaker notion of equivalence, and say that two stable derivations are *weakly equivalent* if they have the same set of stably derivable atoms. Of course, two equivalent stable derivations are also weakly equivalent. Further, similarly to what has been done in Niemelä (2008) for level rankings, we can impose additional restrictions on stable derivations in order to enforce that any two of them are either not weakly equivalent or map $\perp$ to a different ordinal. We do this by introducing strict stable derivations.

**Definition 2.** *A stable derivation $\lambda$ is strict if it maps each atom $A \in V$ to an ordinal $\lambda(A)$ such that $\lambda(A) \leq \lambda(\perp)$ and either $\lambda(A) = 1$ or for each rule $r \in \Pi$ with head $A$,*

1. *either there exists an atom $B \in body^+(r)$ with $\lambda(A) \leq \lambda(B) + 1$, or*
2. *there exists an atom $B \in body^-(r)$ with $\lambda(B) < \lambda(\perp)$.*

We will illustrate the definition of a strict stable derivation on a simple example, which also shows that condition 2. is necessary.

**Example 6.** *Consider the following program:*

$$\begin{gathered} A, \\ B \leftarrow A, \\ B \leftarrow \neg A. \end{gathered} \tag{7}$$

*For these rules, following the conditions of a strict stable derivation, we will have*

$$\begin{gathered} \lambda(A) = 1, \\ \lambda(B) = 1 \text{ or } \lambda(B) \leq \lambda(A) + 1, \\ \lambda(B) = 1 \text{ or } \lambda(A) < \lambda(\perp). \end{gathered} \tag{8}$$

*Without condition 2., we would need to impose $\lambda(B) = 1$, which is not possible due to $\lambda(A) = 1 < \lambda(\bot)$ and $\lambda(B) = 1 \geq \lambda(\bot)$ implied by the definition of a stable derivation.*

Notice also that the above conditions trivially hold when $A = \bot$ and, thus, in a strict stable derivation they hold for each $A \in V^\bot$. Further, for each $A \in V^\bot$ in a strict stable derivation, it is easy to check that $\lambda(A) = 0$ only if $\lambda(\bot) = 0$ and, thus, only if the set of stably derived atoms is empty.

**Theorem 6.** *Let $\Pi$ be a program in the set $V$ of atoms. For any stable derivation $\lambda$ of $\Pi$ there is a strict stable derivation $\lambda_1$ of $\Pi$ which is equivalent to $\lambda$ and such that $\lambda_1(\bot) = |V| + 1$ if $V$ is finite, and $\lambda_1(\bot) = \omega$, otherwise. Any two distinct weakly equivalent strict stable derivations of $\Pi$ differ only in the ordinal associated to $\bot$.*

*Proof.* Let $M$ be the set of atoms stably derived by $\lambda$. Consider the stable derivation $\lambda_1$ having $M$ as stably derived set of atoms constructed in the "left to right" direction of the proof of Theorem 5. By construction $\lambda_1$ is strict, equivalent to $\lambda$ and satisfies $\lambda_1(\bot) = \omega$. On the other hand, it is clear that if $V$ is finite, then the proof still holds if in the proof we replace $M = T_{\Pi^M}\uparrow^\omega$ with $M = T_{\Pi^M}\uparrow^{|V|+1}$ and impose $\lambda_1(\bot) = |V| + 1$.

Assume there are two weakly equivalent strict stable derivations $\lambda_1$ and $\lambda_2$ and an atom $A \in V$ with $\lambda_1(A) \neq \lambda_2(A)$, and either $\lambda_1(A) \neq \lambda_1(\bot)$ or $\lambda_2(A) \neq \lambda_2(\bot)$. Since $\lambda_1$ and $\lambda_2$ are weakly equivalent then $\lambda_1(A) < \lambda_1(\bot)$ and $\lambda_2(A) < \lambda_2(\bot)$. Take such atom $A$ to be such that for each atom $B \in V$ with $\lambda_1(B) \neq \lambda_2(B)$, $min(\lambda_1(A), \lambda_2(A)) \leq min(\lambda_1(B), \lambda_2(B))$. Assume $min(\lambda_1(A), \lambda_2(A)) = \lambda_1(A)$ (analogous proof can be done for the other case). Thus, for each atom $B$ with $\lambda_1(B) < \lambda_1(A)$, $\lambda_1(B) = \lambda_2(B)$. Further, from $\lambda_1(A) < \lambda_2(A)$, it follows $\lambda_2(A) > 1$. Then, $\lambda_1(A) < \lambda_1(\bot)$, $\lambda_2(A) < \lambda_2(\bot)$ and the equivalence between $\lambda_1$ and $\lambda_2$ implies the existence of a rule $r$ with $head(r) = A$ and

1. for each $B \in body^+(r)$, $\lambda_1(B) = \lambda_2(B) < \lambda_1(A) < \lambda_2(A)$,
2. for each $B \in body^-(r)$, $\lambda_1(B) = \lambda_1(\bot)$ and $\lambda_2(B) = \lambda_2(\bot)$, and
3. since $\lambda_2(A) > 1$, there exists some $B \in body^+(r)$ such that $\lambda_1(B)+1 = \lambda_2(B)+1 \geq \lambda_2(A) > \lambda_1(A) \geq \lambda_1(B) + 1$, which is not possible.

□

It is worth observing that our definition of strict stable derivation explicitly imposes that for each atom $A \in V$, $\lambda(A) \leq \lambda(\bot)$. However, $\lambda(A) \leq \lambda(\bot)$ is already entailed by the definition of stable derivation for those atoms $A$ for which the rule $A \leftarrow \bot$ is in $\Pi$.

## 4. A reduction of stable derivations/models to difference logic

The simple definition of (strict) stable derivation has a correspondingly reduction to difference logic, which, thanks to Theorem 5, also characterizes stable models.

Consider a finite program $\Pi$ over a finite set $V$ of variables. In the reduction of $\Pi$ to difference logic, we introduce a variable $\lambda_{dl}(A)$ for each atom $A \in V^\bot$, while the set $\lambda_{dl}(\Pi)$ of formulas corresponding to $\Pi$ contains the formula

$$(\lambda_{dl}(A) < \lambda_{dl}(\bot)) \equiv \bigvee_{r\in\Pi:head(r)=A}\Big(\bigwedge_{B\in body^+(r)}(\lambda_{dl}(A) > \lambda_{dl}(B)) \wedge \bigwedge_{B\in body^-(r)}(\lambda_{dl}(B) \geq \lambda_{dl}(\bot))\Big), \tag{9}$$

for each $A \in V^{\perp}$. For a set $M$ of atoms, let $\lambda_{dl}(M)$ be the following set of formulas in difference logic:

$$\lambda_{dl}(M) = \{\lambda_{dl}(A) < \lambda_{dl}(\perp) \mid A \in M\} \cup \{\lambda_{dl}(A) \geq \lambda_{dl}(\perp) \mid A \notin M\}.$$

**Theorem 7.** *Let $\Pi$ be a finite program. A set $M$ of atoms is a stable model of $\Pi$ or, equivalently, is stably derivable from $\Pi$ iff $\lambda_{dl}(\Pi) \cup \lambda_{dl}(M)$ is satisfiable in difference logic.*

*Proof.* Each formula in $\lambda_{dl}(\Pi)$ is a direct translation of the corresponding condition in the definition of stable derivation.

Thus, for the left to right direction, every stable derivation $\lambda$ corresponds to an interpretation $\sigma$ such that, for each atom $A \in V$ with $\lambda(A) < \lambda(\perp)$, $\sigma(\lambda_{dl}(A)) = \lambda(A)$, and for each atom $A \in V$ with $\lambda(A) \geq \lambda(\perp)$, $\sigma(\lambda_{dl}(A)) = V_{max}$, where $V_{max}$ is a value greater than any value assigned to the variables $A$ with $\lambda(A) < \lambda(\perp)$. $\sigma$ satisfies $\lambda_{dl}(\Pi) \cup \lambda_{dl}(M)$, where $M$ is the set of atoms stably derived by $\lambda$.

For the right to left direction, if $\sigma$ is a model of $\lambda_{dl}(\Pi) \cup \lambda_{dl}(M)$, we can put the set $S$ of values assigned by $\sigma$ in one to one correspondence with the first $|S|$ ordinals respecting the ordering. Then, if $f$ is the function defining the correspondence between the two sets, for each atom $A \in V^{\perp}$, define $\lambda(A) = f(\sigma(\lambda_{dl}(A)))$. By construction, for each $A, B \in V^{\perp}$, $\lambda(A) \leq \lambda(B)$ iff $\sigma(\lambda_{dl}(A)) \leq \sigma(\lambda_{dl}(B))$ and, thus, given the correspondence between $\lambda_{dl}(\Pi)$ and the definition of stable derivation, $\lambda$ is a stable derivation in which $M$ is the set of atoms stably derived by $\lambda$. □

The above theorem allows us to compute stably derivable sets of atoms of a program $\Pi$ with difference logic solvers. Indeed, given a model $\sigma$ of $\lambda_{dl}(\Pi)$, $M = \{A : \sigma(\lambda_{dl}(A)) < \sigma(\lambda_{dl}(\perp))\}$ is a stably derivable set of atoms.

We can also provide the corresponding translation for the additional conditions holding for strict stable derivations. However, difficulties arise if we consider variables ranging over the rationals/reals. In fact, in such cases, when $\lambda_{dl}(A) < \lambda_{dl}(\perp)$ we would like to impose additional conditions forcing $\lambda_{dl}(A)$ to be the "successor" value of some value $\lambda_{dl}(B) < \lambda_{dl}(A)$, and if $\lambda_{dl}(B)$ ranges over the rationals/reals there is no such successive value. Further, difference logic does not allow to impose that a given variable is greater or equal than a constant, and the set of rationals/reals/integers does not have any minimum value.

A simple solution to all the above problems—providing a translation to the observations of the previous section—is to modify condition (9) to

$$\begin{aligned}(\lambda_{dl}(A) < \lambda_{dl}(\perp)) \equiv \textstyle\bigvee_{r\in\Pi:head(r)=A}(\textstyle\bigwedge_{B\in body^+(r)}&(\lambda_{dl}(A) \geq \lambda_{dl}(B)+1) \wedge \\ \textstyle\bigwedge_{B\in body^-(r)}&(\lambda_{dl}(B) \geq \lambda_{dl}(\perp))) \vee \\ (\lambda_{dl}(A) > \lambda_{dl}(\perp)),&\end{aligned} \tag{10}$$

and then include the formula corresponding to the strictness conditions:

$$\begin{aligned}\textstyle\bigwedge_{r\in\Pi:head(r)=A}(&\textstyle\bigvee_{B\in body^+(r)}(\lambda_{dl}(A) \leq \lambda_{dl}(B)+1) \vee \\ &\textstyle\bigvee_{B\in body^-(r)}(\lambda_{dl}(B) < \lambda_{dl}(\perp)) \vee \\ &(\lambda_{dl}(A) = \lambda_{dl}(\top))).\end{aligned} \tag{11}$$

As already the case for the encoding of strong level ranking in difference logic, $\lambda_{dl}(\top)$ is a new variable that is supposed to be interpreted as 1, which is introduced to respect the syntax of difference logic. Let $\lambda_{sdl}(\Pi)$ be the set consisting of the formulas (10) and (11) for each $A \in V^{\perp}$ or $r \in \Pi$, respectively.

**Theorem 8.** *Let $\Pi$ be a program over a finite set $V$ of atoms. For each $A \in V$, $\lambda_{sdl}(\Pi)$ entails both $\lambda_{dl}(A) \leq \lambda_{dl}(\perp)$ as well as $\lambda_{dl}(A) = \lambda_{dl}(\perp)$ if $\lambda_{dl}(\top) \geq \lambda_{dl}(\perp)$.*

*Proof.* $\lambda_{dl}(A) \leq \lambda_{dl}(\perp)$ is an easy consequence of (10). Assume there is a model $\sigma$ of $\lambda_{sdl}(\Pi)$ such that $\sigma(\lambda_{dl}(\top)) \geq \sigma(\lambda_{dl}(\perp))$ and $\sigma(\lambda_{dl}(A)) < \sigma(\lambda_{dl}(\perp))$ for some variable $\lambda_{dl}(A)$. Consider such a variable to be one with minimum $\sigma(\lambda_{dl}(A))$ value. Since $\sigma(\lambda_{dl}(A)) < \sigma(\lambda_{dl}(\perp))$, by (10) there must be a rule $r$ with head $A$, $body^{+}(r) = \emptyset$ (otherwise, $\sigma(\lambda_{dl}(B)) < \sigma(\lambda_{dl}(A))$ for each $B \in body^{+}(r)$ contradicts that $\lambda_{dl}(A)$ is a variable with minimum $\sigma(\lambda_{dl}(A))$ value), and $\sigma(\lambda_{dl}(B)) = \sigma(\lambda_{dl}(\perp))$ for each $B \in body^{-}(r)$. Then, by (11), $\sigma(\lambda_{dl}(A)) = \sigma(\lambda_{dl}(\top))$. But $\sigma(\lambda_{dl}(\top)) = \sigma(\lambda_{dl}(A)) < \sigma(\lambda_{dl}(\perp))$, which contradicts the other initial hypothesis that $\sigma(\lambda_{dl}(\top)) \geq \sigma(\lambda_{dl}(\perp))$. □

**Theorem 9.** *Let $\Pi$ be a program over a finite set $V$ of atoms. Let $\sigma$ be an interpretation of $\lambda_{sdl}(\Pi)$ such that $\sigma(\lambda_{dl}(\top)) = 1$ and $\sigma(\lambda_{dl}(\perp)) = |V| + 1$. Let $\lambda$ be the function such that, for each $A \in V^{\perp}$, $\lambda(A) = \sigma(\lambda_{dl}(A))$. Then, $\sigma$ is a model of $\lambda_{sdl}(\Pi)$ iff $\lambda$ is a strict stable derivation.*

*Proof.* Formulas (10) and (11) are a direct translation of the corresponding condition in the definition of strict stable derivation. □

**Example 7.** *Let $\Pi$ be the set of rules in (2). The formulas (9) in $\lambda_{dl}(\Pi)$ are*

$$(\lambda_{dl}(A) < \lambda_{dl}(\perp)) \equiv (\lambda_{dl}(A) > \lambda_{dl}(B)) \vee (\lambda_{dl}(C) \geq \lambda_{dl}(\perp)), \tag{12}$$
$$(\lambda_{dl}(B) < \lambda_{dl}(\perp)) \equiv (\lambda_{dl}(B) > \lambda_{dl}(A)), \tag{13}$$
$$(\lambda_{dl}(C) < \lambda_{dl}(\perp)) \equiv (\lambda_{dl}(C) > \lambda_{dl}(C)), \tag{14}$$
$$(\lambda_{dl}(\perp) < \lambda_{dl}(\perp)) \equiv \perp. \tag{15}$$

*Any model of these formulas satisfies $\lambda_{dl}(A) < \lambda_{dl}(B) < \lambda_{dl}(\perp) \leq \lambda_{dl}(C)$.*

*The formulas (10) and (11) in $\lambda_{sdl}(\Pi)$, encoding strict stable derivations, are*

$$\begin{gathered}
(\lambda_{dl}(A) < \lambda_{dl}(\perp)) \equiv (\lambda_{dl}(A) \geq \lambda_{dl}(B) + 1) \vee (\lambda_{dl}(C) \geq \lambda_{dl}(\perp)) \vee (\lambda_{dl}(A) > \lambda_{dl}(\perp)),\\
(\lambda_{dl}(B) < \lambda_{dl}(\perp)) \equiv (\lambda_{dl}(B) \geq \lambda_{dl}(A) + 1) \vee (\lambda_{dl}(B) > \lambda_{dl}(\perp)),\\
(\lambda_{dl}(C) < \lambda_{dl}(\perp)) \equiv (\lambda_{dl}(C) \geq \lambda_{dl}(C) + 1) \vee (\lambda_{dl}(C) > \lambda_{dl}(\perp)),\\
(\lambda_{dl}(\perp) < \lambda_{dl}(\perp)) \equiv (\lambda_{dl}(\perp) > \lambda_{dl}(\perp)),\\
(\lambda_{dl}(A) \leq \lambda_{dl}(B) + 1) \vee (\lambda_{dl}(A) = \lambda_{dl}(\top)),\\
(\lambda_{dl}(C) < \lambda_{dl}(\perp)) \vee (\lambda_{dl}(A) = \lambda_{dl}(\top)),\\
(\lambda_{dl}(B) \leq \lambda_{dl}(A) + 1) \vee (\lambda_{dl}(B) = \lambda_{dl}(\top)),\\
(\lambda_{dl}(C) \leq \lambda_{dl}(C) + 1) \vee (\lambda_{dl}(C) = \lambda_{dl}(\top)),
\end{gathered}$$

*and they entail $\lambda_{dl}(\top) = \lambda_{dl}(A) < \lambda_{dl}(A) + 1 = \lambda_{dl}(B) < \lambda_{dl}(\perp) = \lambda_{dl}(C)$.*

The proposed encoding in difference logic of (strict) stable derivations has thus $|V| + 1$ numeric variables and size $O(||\Pi||)$, while Niemelä's encoding (2008) of (strong) level ranking includes $|V|$ Boolean and $|V|$ numeric variables. We thus provide a linear encoding, in the size of a program, in a fragment of classical first-order logic (difference logic).

## 5. Improved reduction by handling SCCs

The SCCs of the positive dependency graph, introduced in Section 2, associated with a program $\Pi$ partition the atoms $V$ into equivalence classes such that the members of each class mutually reach one another by (possibly empty) paths. We refer to the partition of $V$ based on SCCs by $\mathrm{SCC}(\Pi)$ and by $\mathrm{SCC}(A)$ we denote the part including the atom $A \in V$. Moreover, any injective function $\tau : \mathrm{SCC}(\Pi) \mapsto \mathbb{N}$ such that $\tau(\mathrm{SCC}(A_1)) \geq \tau(\mathrm{SCC}(A_2))$ holds for each edge from $A_1$ to $A_2$ in the positive dependency graph of $\Pi$ is a *topological ordering* of $\mathrm{SCC}(\Pi)$. Some topological ordering $\tau$ of $\mathrm{SCC}(\Pi)$ is guaranteed to exist, while $\tau$ is in general not unique even if $\tau(\mathrm{SCC}(A)) \geq 1$ for the atoms $A \in V$ are required to be successive natural numbers.

The intuitive role of $\mathrm{SCC}(A)$ is to gather the atoms that may contribute to (unstable) derivations in which $A$ circularly supports itself. This in turn means that atoms outside $\mathrm{SCC}(A)$ cannot undermine the stability of a (supported) model $M$ of $\Pi$, as formally stated by Theorem 1. Accordingly, the following refined difference logic reduction of (strict) stable derivations exploits SCCs to condition $\lambda_{dl}(A)$ values differing from $\lambda_{dl}(\bot)$ and $\lambda_{dl}(\top)$ on atoms in $\mathrm{SCC}(A)$ only.

First, for a program $\Pi$ over a finite set $V$ of atoms, the formula (9) can be updated as follows:

$$\begin{aligned}(\lambda_{dl}(A) < \lambda_{dl}(\bot)) \equiv \textstyle\bigvee_{r\in\Pi:head(r)=A}(&\textstyle\bigwedge_{B\in body^+(r)\cap \mathrm{SCC}(A)}(\lambda_{dl}(A) > \lambda_{dl}(B)) \wedge \\ &\textstyle\bigwedge_{B\in body^+(r)\setminus \mathrm{SCC}(A)}(\lambda_{dl}(B) < \lambda_{dl}(\bot)) \wedge \\ &\textstyle\bigwedge_{B\in body^-(r)}(\lambda_{dl}(B) \geq \lambda_{dl}(\bot))),\end{aligned} \tag{16}$$

where the condition $\lambda_{dl}(B) < \lambda_{dl}(\bot)$ is used for atoms $B$ in the positive part of a rule body whenever $B$ does not belong to $\mathrm{SCC}(A)$. We thus merely inspect the value $\lambda_{dl}(B)$ but do not relate it to $\lambda_{dl}(A)$, just alike the dual condition $\lambda_{dl}(B) \geq \lambda_{dl}(\bot)$ for atoms $B$ in the negative part of a rule body.

**Example 8.** *For the program $\Pi$ consisting of the rules in (2), we have that $\mathrm{SCC}(\Pi) = \{\{A,B\},\{C\}\}$, and the formulas (16) in $\lambda_{dl}(\Pi)$ are (12)–(15) as in Example 7. Moreover, the SCCs remain unchanged when the rule $A \leftarrow \neg C$ is replaced by $A \leftarrow C$, in which case (12) turns into*

$$(\lambda_{dl}(A) < \lambda_{dl}(\bot)) \equiv (\lambda_{dl}(A) > \lambda_{dl}(B)) \vee (\lambda_{dl}(C) < \lambda_{dl}(\bot)),$$

*including $\lambda_{dl}(C) < \lambda_{dl}(\bot)$ instead of $\lambda_{dl}(A) > \lambda_{dl}(C)$ for the positive body atom $C$ from outside $\mathrm{SCC}(A) = \{A,B\}$. Either formulation of $\lambda_{dl}(\Pi)$ for the modified program $\Pi$ entails $\lambda_{dl}(\bot) \leq \lambda_{dl}(C)$ and $\lambda_{dl}(\bot) \leq \lambda_{dl}(A) = \lambda_{dl}(B)$, which means that neither $A$, $B$, nor $C$ can be stably derived.*

In general, Theorem 7 remains valid when $\lambda_{dl}(\Pi)$ is based on the formula (16) instead of (9). The left to right direction in the proof of Theorem 7 still applies, and for the right to left direction, we use some topological ordering $\tau$ of $\mathrm{SCC}(\Pi)$ to modify a model $\sigma'$ of $\lambda_{dl}(\Pi) \cup \lambda_{dl}(M)$:

$$\sigma(\lambda_{dl}(A)) = \begin{cases} \max\{\sigma(\lambda_{dl}(B)) : B \in M, \tau(\mathrm{SCC}(B)) < \tau(\mathrm{SCC}(A))\} & \\ \qquad + |\{B \in \mathrm{SCC}(A) : \sigma'(\lambda_{dl}(B)) \leq \sigma'(\lambda_{dl}(A))\}| & \text{if } A \in M, \\ \max\{\sigma(\lambda_{dl}(B)) : B \in M\} + 1 & \text{if } A \notin M. \end{cases}$$

This inductive remapping leads to $\sigma(\lambda_{dl}(A)) = \sigma(\lambda_{dl}(\bot))$ iff $\sigma'(\lambda_{dl}(A)) \geq \sigma'(\lambda_{dl}(\bot))$, and in addition, $\sigma'(\lambda_{dl}(B)) < \sigma'(\lambda_{dl}(\bot))$ yields $\sigma(\lambda_{dl}(A)) > \sigma(\lambda_{dl}(B))$ for every atom $B \notin \mathrm{SCC}(A)$ occurring in the positive body part of some rule $r \in \Pi$ with $\mathit{head}(r) = A$. As a consequence, the model $\sigma$ of $\lambda_{dl}(\Pi) \cup \lambda_{dl}(M)$ reestablishes the right to left argument in the proof of Theorem 7.

Second, the formulas (10) and (11) for the strictness conditions can be updated to incorporate SCCs:

$$\begin{aligned}(\lambda_{dl}(A) < \lambda_{dl}(\bot)) \equiv \textstyle\bigvee_{r\in\Pi:\mathit{head}(r)=A}(&\textstyle\bigwedge_{B\in \mathit{body}^+(r)\cap\mathrm{SCC}(A)}(\lambda_{dl}(A) \geq \lambda_{dl}(B)+1) \wedge \\ &\textstyle\bigwedge_{B\in \mathit{body}^+(r)\setminus\mathrm{SCC}(A)}(\lambda_{dl}(B) < \lambda_{dl}(\bot)) \wedge \\ &\textstyle\bigwedge_{B\in \mathit{body}^-(r)}(\lambda_{dl}(B) \geq \lambda_{dl}(\bot))) \vee \\ (\lambda_{dl}(A) > \lambda_{dl}(\bot))&,\end{aligned} \tag{17}$$

$$\begin{aligned}\textstyle\bigwedge_{r\in\Pi:\mathit{head}(r)=A}(&\textstyle\bigvee_{B\in \mathit{body}^+(r)\cap\mathrm{SCC}(A)}(\lambda_{dl}(A) \leq \lambda_{dl}(B)+1) \vee \\ &\textstyle\bigvee_{B\in \mathit{body}^+(r)\setminus\mathrm{SCC}(A)}(\lambda_{dl}(B) \geq \lambda_{dl}(\bot)) \vee \\ &\textstyle\bigvee_{B\in \mathit{body}^-(r)}(\lambda_{dl}(B) < \lambda_{dl}(\bot)) \vee \\ &(\lambda_{dl}(A) = \lambda_{dl}(\top))).\end{aligned} \tag{18}$$

The expression $\lambda_{dl}(B) < \lambda_{dl}(\bot)$ or $\lambda_{dl}(B) \geq \lambda_{dl}(\bot)$, respectively, again inspects the value $\lambda_{dl}(B)$ for atoms $B$ in the positive part of a rule body, but does not relate $\lambda_{dl}(B)$ to $\lambda_{dl}(A)$ whenever $B$ does not belong to $\mathrm{SCC}(A)$. The arguments in the proof of Theorem 8 remain valid when $\lambda_{sdl}(\Pi)$ consists of the formulas (17) and (18) instead of (10) and (11). For a counterpart of Theorem 9 on the correspondence between model $\sigma$ of $\lambda_{sdl}(\Pi)$ and strict stable derivations, we associate $\sigma$ with the inductively defined function

$$\lambda(A) = \begin{cases} \min\{\max\{\lambda(B) \mid B \in \mathit{body}^+(r)\} \mid \\ \quad r \in \Pi, \{B \in \mathit{body}^+(r) \cap \mathrm{SCC}(A) \mid \sigma(\lambda_{dl}(B)) \geq \sigma(\lambda_{dl}(A))\} \cup \\ \qquad \{B \in \mathit{body}^+(r) \setminus \mathrm{SCC}(A) \mid \sigma(\lambda_{dl}(B)) = \sigma(\lambda_{dl}(\bot))\} \cup \\ \qquad \{B \in \mathit{body}^-(r) \mid \sigma(\lambda_{dl}(B)) < \sigma(\lambda_{dl}(\bot))\} = \emptyset\} + 1 \\ \qquad\quad \text{if } \sigma(\lambda_{dl}(A)) < \sigma(\lambda_{dl}(\bot)), \\ \lambda(\bot) \qquad\quad \text{if } \sigma(\lambda_{dl}(A)) = \sigma(\lambda_{dl}(\bot)). \end{cases}$$

This function $\lambda$ is a strict stable derivation iff $\sigma$ is a model of $\lambda_{sdl}(\Pi)$ based on the formulas (17) and (18), so that Theorem 9 carries forward to strictness conditions refined by SCCs.

**Example 9.** *Let* $\Pi$ *be the set of rules in* (2) *augmented with* $B \leftarrow C$ *and* $C \leftarrow \neg D$, *for which we obtain* $\mathrm{SCC}(\Pi) = \{\{A,B\},\{C\},\{D\}\}$. *The formulas* (17) *in* $\lambda_{sdl}(\Pi)$ *are*

$$\begin{aligned}(\lambda_{dl}(A) < \lambda_{dl}(\bot)) &\equiv (\lambda_{dl}(A) \geq \lambda_{dl}(B)+1) \vee \\ &\quad (\lambda_{dl}(C) \geq \lambda_{dl}(\bot)) \vee (\lambda_{dl}(A) > \lambda_{dl}(\bot)), &(19)\\ (\lambda_{dl}(B) < \lambda_{dl}(\bot)) &\equiv (\lambda_{dl}(B) \geq \lambda_{dl}(A)+1) \vee \\ &\quad (\lambda_{dl}(C) < \lambda_{dl}(\bot)) \vee (\lambda_{dl}(B) > \lambda_{dl}(\bot)), &(20)\\ (\lambda_{dl}(C) < \lambda_{dl}(\bot)) &\equiv (\lambda_{dl}(C) \geq \lambda_{dl}(C)+1) \vee \\ &\quad (\lambda_{dl}(D) \geq \lambda_{dl}(\bot)) \vee (\lambda_{dl}(C) > \lambda_{dl}(\bot)), &(21)\\ (\lambda_{dl}(D) < \lambda_{dl}(\bot)) &\equiv (\lambda_{dl}(D) > \lambda_{dl}(\bot)), &(22)\\ (\lambda_{dl}(\bot) < \lambda_{dl}(\bot)) &\equiv (\lambda_{dl}(\bot) > \lambda_{dl}(\bot)). &(23)\end{aligned}$$

*Taken on their own,* (19)–(23) *entail* $\lambda_{dl}(B) + 1 \leq \lambda_{dl}(A) < \lambda_{dl}(\bot)$ *and* $\lambda_{dl}(C) < \lambda_{dl}(\bot) = \lambda_{dl}(D)$*. Along with the strictness conditions* (18) *obtained per rule, which are*

$$\begin{aligned}
(\lambda_{dl}(A) \leq \lambda_{dl}(B) + 1) \vee (\lambda_{dl}(A) = \lambda_{dl}(\top)),\\
(\lambda_{dl}(C) < \lambda_{dl}(\bot)) \vee (\lambda_{dl}(A) = \lambda_{dl}(\top)),\\
(\lambda_{dl}(B) \leq \lambda_{dl}(A) + 1) \vee (\lambda_{dl}(B) = \lambda_{dl}(\top)),\\
(\lambda_{dl}(C) \geq \lambda_{dl}(\bot)) \vee (\lambda_{dl}(B) = \lambda_{dl}(\top)),\\
(\lambda_{dl}(C) \leq \lambda_{dl}(C) + 1) \vee (\lambda_{dl}(C) = \lambda_{dl}(\top)),\\
(\lambda_{dl}(D) < \lambda_{dl}(\bot)) \vee (\lambda_{dl}(C) = \lambda_{dl}(\top)),
\end{aligned}$$

$\lambda_{sdl}(\Pi)$ *entails* $\lambda_{dl}(\top) = \lambda_{dl}(C) = \lambda_{dl}(B) < \lambda_{dl}(B) + 1 = \lambda_{dl}(A) < \lambda_{dl}(\bot) = \lambda_{dl}(D)$*. The resulting model of* $\lambda_{sdl}(\Pi)$ *represents the strict stable derivation* $\lambda(C) = 1, \lambda(B) = 2, \lambda(A) = 3, \lambda(D) = \lambda(\bot)$*.*

We note that difference logic expressions in $\lambda_{dl}(\Pi)$ as well as $\lambda_{sdl}(\Pi)$ are solely of the form $\lambda_{dl}(A) < \lambda_{dl}(\bot)$, $\lambda_{dl}(A) \geq \lambda_{dl}(\bot)$, $\lambda_{dl}(A) > \lambda_{dl}(\bot)$, or $\lambda_{dl}(A) = \lambda_{dl}(\top)$ when $\Pi$ is tight, which resembles how existing encodings take advantage of SCCs to dismiss atom levels and default to Clark's completion for tight programs (Janhunen, 2004; Niemelä, 2008; Janhunen et al., 2009; Gebser et al., 2014).

## 6. Implementation and Experimental Analysis

In this section, we present details about the implementation of our tool, the benchmarks employed for the evaluation, and the results of our experiments, in three separate subsections. First, we describe details of our implementation, which allows us to use SMT solvers starting from the characterization in previous sections. Then, we describe the benchmarks that we used for testing the translation, together with our experimental settings. Finally, the results that we obtained are presented.

### *6.1. Implementation*

In this subsection we first present implementation details that allowed us to represent our formulas in a format such that SMT solvers can be used on the reduction provided in Section 4. Then, we show how to include the improvements due to SCCs from Section 5. Finally, we present our tool ASP2IDL.

The encoding is based on the formula (9), adapted and optimized to be used with an SMT solver.

In particular, given a program $\Pi$, starting from formula (9) we can have a direct translation.

For every rule $r$, with $head(r) = A$, we add a formula of the form:

$$\lambda_{dl}(A) < \lambda_{dl}(\bot) \equiv (\bigwedge_{B \in body^+(r)} (\lambda_{dl}(A) > \lambda_{dl}(B)) \wedge \bigwedge_{B \in body^-(r)} \neg(\lambda_{dl}(B) < \lambda_{dl}(\bot))).$$

In order to optimize the translation to difference logic, we decided to split such a formula. Thus, for the i-th rule $r_i$ of $\Pi$ we introduce a new Boolean variable denoted with $w_i$ and, assuming $head(r_i) = A$, we add a formula of the form:

$$w_i \to \big(\bigwedge_{B\in body^+(r_i)}(\lambda_{dl}(A) > \lambda_{dl}(B))\wedge \\ \bigwedge_{B\in body^-(r_i)} \neg(\lambda_{dl}(B) < \lambda_{dl}(\bot))\big). \tag{24}$$

At the same time, we consider all the formulas in which $A$ is in the head of a rule, and add:

$$\lambda_{dl}(A) < \lambda_{dl}(\bot) \to \bigvee_{r_i\in\Pi:head(r_i)=A} w_i. \tag{25}$$

Further, we can add, for every rule $r$ with head $A$ the formula:

$$\big(\bigwedge_{B\in body^+(r)}(\lambda_{dl}(B) < \lambda_{dl}(\bot))\wedge \bigwedge_{B\in body^-(r)} \neg(\lambda_{dl}(B) < \lambda_{dl}(\bot))\big) \to \lambda_{dl}(A) < \lambda_{dl}(\bot). \tag{26}$$

Finally, it is easy to prove that, for each atom $A$ which is a head of a rule and for each atom $B$ occurring in the positive body of the rule with head $A \neq \bot$, a formula of the form:

$$\lambda_{dl}(A) > \lambda_{dl}(B) \to \lambda_{dl}(\bot) > \lambda_{dl}(B) \tag{27}$$

can be safely added still obtaining an equisatisfiable reduction.

**Example 10.** *Consider the program $\Pi$ in the four atoms $A, B, C, D$ whose rules are:*

$$\begin{array}{c} A \leftarrow B, \\ A \leftarrow C, \\ B \leftarrow A, \\ B \leftarrow C, \neg D, \\ C \leftarrow A, B, \\ C \leftarrow \neg D, \\ D \leftarrow \neg C. \end{array} \tag{28}$$

*Such program would be translated, following formulas (24), (25), (26), and (27) into:*

$$\begin{gathered}
w_1 \rightarrow \lambda_{dl}(B) < \lambda_{dl}(A), \\
w_2 \rightarrow \lambda_{dl}(C) < \lambda_{dl}(A), \\
w_3 \rightarrow \lambda_{dl}(A) < \lambda_{dl}(B), \\
w_4 \rightarrow \lambda_{dl}(C) < \lambda_{dl}(B) \wedge \neg(\lambda_{dl}(D) < \lambda_{dl}(\bot)), \\
w_5 \rightarrow \lambda_{dl}(A) < \lambda_{dl}(C) \wedge \lambda_{dl}(B) < \lambda_{dl}(C), \\
w_6 \rightarrow \neg(\lambda_{dl}(D) < \lambda_{dl}(\bot)), \\
w_7 \rightarrow \neg(\lambda_{dl}(C) < \lambda_{dl}(\bot)),
\end{gathered}$$

$$\begin{gathered}
\lambda_{dl}(A) < \lambda_{dl}(\bot) \rightarrow (w_1 \vee w_2), \\
\lambda_{dl}(B) < \lambda_{dl}(\bot) \rightarrow (w_3 \vee w_4), \\
\lambda_{dl}(C) < \lambda_{dl}(\bot) \rightarrow (w_5 \vee w_6), \\
\lambda_{dl}(D) < \lambda_{dl}(\bot) \rightarrow w_7,
\end{gathered}$$

$$\begin{gathered}
\lambda_{dl}(B) < \lambda_{dl}(\bot) \rightarrow \lambda_{dl}(A) < \lambda_{dl}(\bot), \\
\lambda_{dl}(C) < \lambda_{dl}(\bot) \rightarrow \lambda_{dl}(A) < \lambda_{dl}(\bot), \\
\lambda_{dl}(A) < \lambda_{dl}(\bot) \rightarrow \lambda_{dl}(B) < \lambda_{dl}(\bot), \\
(\lambda_{dl}(C) < \lambda_{dl}(\bot) \wedge \neg(\lambda_{dl}(D) < \lambda_{dl}(\bot))) \rightarrow \lambda_{dl}(B) < \lambda_{dl}(\bot), \\
(\lambda_{dl}(A) < \lambda_{dl}(\bot) \wedge \lambda_{dl}(B) < \lambda_{dl}(\bot)) \rightarrow \lambda_{dl}(C) < \lambda_{dl}(\bot), \\
\neg(\lambda_{dl}(D) < \lambda_{dl}(\bot)) \rightarrow \lambda_{dl}(C) < \lambda_{dl}(\bot), \\
\neg(\lambda_{dl}(C) < \lambda_{dl}(\bot)) \rightarrow \lambda_{dl}(D) < \lambda_{dl}(\bot),
\end{gathered}$$

$$\begin{gathered}
\lambda_{dl}(B) < \lambda_{dl}(A) \rightarrow \lambda_{dl}(B) < \lambda_{dl}(\bot), \\
\lambda_{dl}(C) < \lambda_{dl}(A) \rightarrow \lambda_{dl}(C) < \lambda_{dl}(\bot), \\
\lambda_{dl}(A) < \lambda_{dl}(B) \rightarrow \lambda_{dl}(A) < \lambda_{dl}(\bot), \\
\lambda_{dl}(C) < \lambda_{dl}(B) \rightarrow \lambda_{dl}(C) < \lambda_{dl}(\bot), \\
\lambda_{dl}(A) < \lambda_{dl}(C) \rightarrow \lambda_{dl}(A) < \lambda_{dl}(\bot), \\
\lambda_{dl}(B) < \lambda_{dl}(C) \rightarrow \lambda_{dl}(B) < \lambda_{dl}(\bot),
\end{gathered}$$

**Handling SCCs.** In case the starting encoding is non-tight, and thus there are SCCs, some changes are in order. Consider formula (16), given a program $\Pi$, we can rewrite formula (24). For every rule $r_i$, with A $= head(r_i)$, we would rewrite (24) as:

$$\begin{aligned} w_i \rightarrow \ & \big(\textstyle\bigwedge_{B \in body^+(r_i) \cap SCC(A)} \lambda_{dl}(A) > \lambda_{dl}(B) \wedge \\ & \textstyle\bigwedge_{B \in body^+(r_i) \setminus SCC(A)} \lambda_{dl}(B) < \lambda_{dl}(\bot) \wedge \\ & \textstyle\bigwedge_{B \in body^-(r_i)} \neg(\lambda_{dl}(B) < \lambda_{dl}(\bot))\big) \end{aligned} \tag{29}$$

Moreover, it is possible to reduce the number of generated formulas by using formula (27) only for atoms in the same SCC.

The implementation details presented above can be adapted to the strict stable derivation presented in Sections 4 and 5. We remind that, in order to implement the "strict" version, we would have to implement the formulas (10) and (11) instead of the formula (9) for the implementation without the handling of SCCs. In the implementation handling SCCs, formula (16) should be replaced by the implementation of the formulas (17) and (18).

**ASP2IDL System.** To obtain the translated formulas we developed the system ASP2IDL. The system makes use of the Python library PySMT (Gario and Micheli, 2015) to obtain

| Domain | Variables | Translation | Min | Q1 | Median | Q3 | Max |
|---|---|---|---|---|---|---|---|
| Graph Coloring | 12454 | ASP2IDL | 9561 | 9998 | 11137 | 12012 | 12373 |
| | | ASP2IDL+SCC | 5767 | 6025 | 6675 | 7192 | 7373 |
| Hanoi Tower | 146467 | ASP2IDL | 119134 | 146492 | 152440 | 167909 | 260394 |
| | | ASP2IDL+SCC | 72718 | 89421 | 93053 | 102494 | 158848 |
| Visit all | 752263 | ASP2IDL | 23881 | 27912 | 88938 | 112405 | 156705 |
| | | ASP2IDL+SCC | 18874 | 22039 | 70045 | 88487 | 123302 |
| Labyrinth | 325405 | ASP2IDL | 77846 | 128824 | 298583 | 400842 | 459934 |
| | | ASP2IDL+SCC | 60129 | 99431 | 230203 | 308937 | 354455 |
| Knight Tour with Holes | 899753 | ASP2IDL | 113822 | 219744 | 412418 | 533257 | 745260 |
| | | ASP2IDL+SCC | 100550 | 177101 | 314725 | 469446 | 658232 |

Table 1: Average number of variables and minimum, first quartile, median, third quartile, and maximum sizes of the generated formulas obtained by the translations.

the difference logic formulas. The input of the system is a ground instance, obtained through the usage of the ASP grounder GRINGO (Gebser et al., 2019), in the aspif format, which also computes the SCCs in case the domain is non-tight, via the option –REIFY-SCCS. The output of our system is a formula in the SMT-LIB standard language (Barrett et al., 2017), which is fed to the underlying SMT solver. The translation tool is available at https://github.com/MarcoMochi/ASP2IDL.

## *6.2. Benchmarks and Experimental Settings*

We tested our implementation on 6 well-known domains (coming from the 2013 (Calimeri et al., 2014) and 2014 Calimeri et al. (2016) ASP Competitions), used in the 2017 ASP competition (Gebser et al., 2020) falling in Track #1, which correspond to all domains which have encodings that use just normal rules, so our translation can be directly applied to them: Graph Coloring, Hanoi Tower, Knight Tour with Holes, Labyrinth, Stable Marriage and Visit-all. The Graph Coloring, Hanoi Tower, Stable Marriage and Visit-all encodings are tight, while the Knight Tour with Holes and Labyrinth domains are non-tight and thus contain SCCs. We tested the instances using the translation without taking into account SCCs handling (ASP2IDL) and with the SCCs handling enabled (ASP2IDL+SCC), and compared the obtained results to another translation-based solver LP2DIFF version 1.27 (Janhunen et al., 2009), and CLINGO version 5.6.2 as a reference. After preliminary tests, we omitted from the translation the formulas corresponding to strictness conditions, given they do not bring to computational advantages. The instances considered were the ones actually "evaluated" at the competition, and the baseline SMT solver employed was Yices 2 (Dutertre, 2014b). Table 1 gives an indication about the sizes of the translations, by means of the 5-numbers statistics for the domains evaluated (but for Stable Marriage, due to the sizes of the ground ASP programs). The 5 numbers appear in the last columns, while the first three columns of the table report the domain, the number of variables, and the translation, respectively. Detailed statistics can be found in Appendix, and all the resulting SMT formulas can be found at https://github.com/MarcoMochi/ASP2IDL. The tests were performed using a MAC M1 Pro machine, with 3.2GHz and 8 GB of RAM. Each solver was run imposing single-thread. The timeout was set to 600 seconds.

| Instance | ASP2IDL | ASP2IDL+SCC | LP2DIFF | CLINGO | SAT? |
|---|---|---|---|---|---|
| 0002 | - | - | - | - | UNSAT |
| 0004 | 71 | 89 | 30 | **6** | SAT |
| 0006 | 7 | 4 | 13 | **3** | SAT |
| 0011 | - | - | - | - | – |
| 0012 | - | - | - | **307** | SAT |
| 0015 | 268 | 183 | 240 | **56** | UNSAT |
| 0020 | 168 | **90** | - | - | – |
| 0027 | 34 | 20 | 26 | **10** | UNSAT |
| 0030 | 114 | 85 | 29 | **14** | SAT |
| 0032 | 13 | 5 | 10 | **4** | UNSAT |
| 0034 | 151 | 106 | 115 | **33** | UNSAT |
| 0038 | 120 | 94 | 113 | **30** | UNSAT |
| 0043 | 254 | 266 | 290 | **62** | UNSAT |
| 0045 | - | 593 | - | **134** | UNSAT |
| 0051 | - | - | - | **159** | UNSAT |
| 0052 | - | - | - | - | – |
| 0055 | - | - | - | **151** | UNSAT |
| 0056 | - | - | - | **260** | UNSAT |
| 0057 | 16 | 14 | 17 | **5** | UNSAT |
| 0058 | - | - | - | **262** | UNSAT |
| Solved Instances | 11 | 12 | 10 | 16 | |

Table 2: Time (seconds) required to solve the instances in the Graph Coloring domain. A – means that the solver (YICES or CLINGO) was not able to find a solution in 600 seconds.

## 6.3. *Results*

This subsection presents the results of our experimental analysis. First, results of tight domains are shown, then we move to the non-tight domains, in two separate paragraphs. For each domain, we present a table in which the results of the evaluated systems presented in terms of CPU time to solve each instance, rounded to the upper integer number, or "–" for timeout, are compared on the instances evaluated at the competition, highlighting the best result for each instance in bold. Then, a cumulative plot for translation-based approaches is shown, in which solved instances are ordered by solving times, and the time associated to an instance is the sum of the times for solving all easier instances, plus the actual.

*Tight domains.* In Table 2 we start by analyzing Graph Coloring. The table is organized as follows: the first column contains the instance, the columns from the second to the forth contain the results of the translation-based systems, in seconds, for solving the generated SMT formulas. The fifth column contains the results obtained by CLINGO, as a reference, while the last column reports whether the instance is satisfiable or not. The last row contains the total number of instances solved within the time limit. From the table, we can see that ASP2IDL+SCC solves 1 instance more than ASP2IDL, which in turn solves one instance more than LP2DIFF. The difference between ASP2IDL+SCC and ASP2IDL can be ascribed to the smaller number of rules created, as can be seen from Table 1 (see Appendix for full details). CLINGO, instead, solves 4 instances more than the best translation-based approach. The cumulative plot in Figure 1 shows that LP2DIFF is slighly faster in solving its easier instances, but then it solves 1-2 less instances then the ASP2IDL-based approaches.

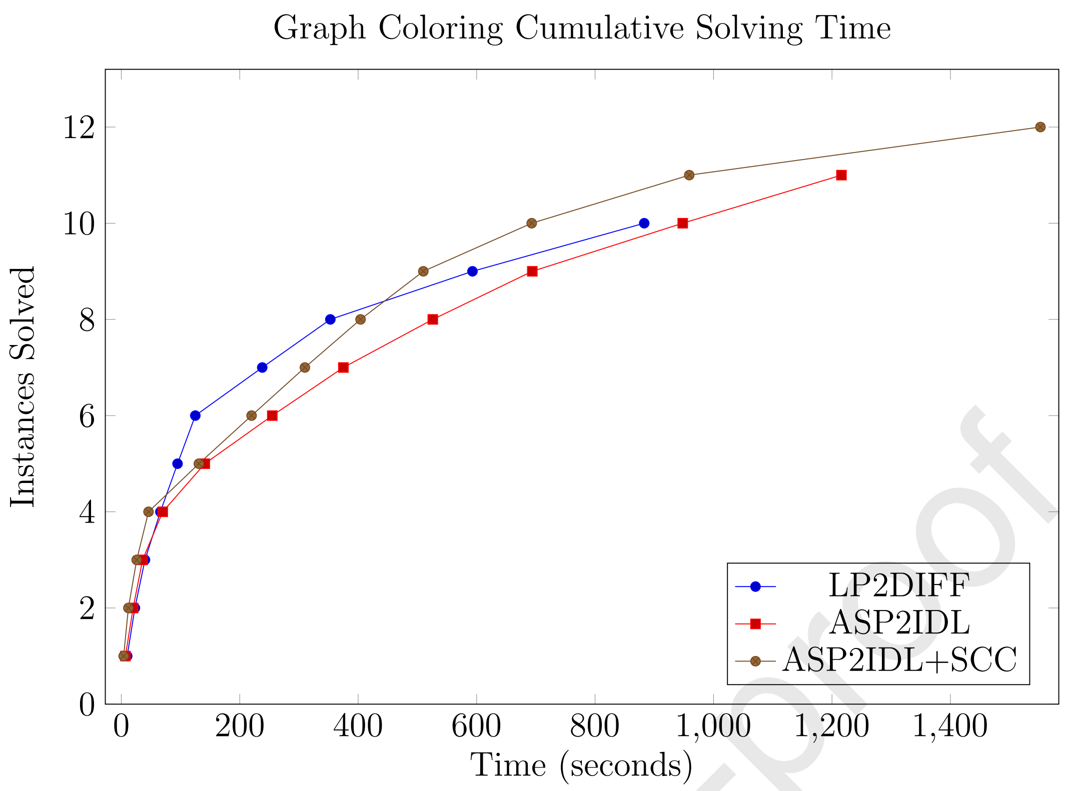


Figure 1: Cumulative time (seconds) required to solve the instances in the Graph Coloring domain by the ASP2IDL, ASP2IDL+SCC and LP2DIFF systems.

Table 3 then presents the results for the Hanoi Tower domain, in which all instances are satisfiable, thus the last column is omitted (and this will be the case for all the remaining domains). It can be noted that the three translation-based systems solve the same instances. Having a look, instead, at Figure 2 which contains cumulative times, it can be noted that ASP2IDL is the solver that takes the larger times for solving its easier instances, while then the picture changes for the hardest instances solved, where it is ASP2IDL+SCC which takes more time.

CLINGO is able to solve 3 additional instances than the translation-based approaches.

The third tight domain analyzed is Visit-all. The results on the Visit-all domain can be seen in Table 4 and are similar to the previous domain, given that all translation-based solvers solve the same instances. However, here, as can be grasped from Figure 3, the version ASP2IDL+SCC is faster. Similarly to the previous domain, CLINGO has better performance compared to the translation-based systems, solving 6 additional instances, though most of them are solved very close to the time limit.

Regarding the Stable Marriage domain, we were not able to obtain any solution with the translation-based approaches, due to the very large size of the ground instances as mentioned earlier. CLINGO can solve only one instance.

*Non-tight domains.* We consider now the non-tight domains. Regarding Labyrinth, the advantages of our system implementing the SCCs management emerge. From Table 5, it can be noted that the ASP2IDL+SCC system is able to solve all the 20 tested instances, while ASP2IDL and LP2DIFF systems can solve only 8 and 6 instances, respectively. Moreover, the time required by ASP2IDL+SCC to solve the instances is less than 10

| Instance | ASP2IDL | ASP2IDL+SCC | LP2DIFF | CLINGO |
|---|---|---|---|---|
| 0001 | 432 | 79 | 96 | **21** |
| 0002 | 60 | 78 | **3** | 4 |
| 0003 | 27 | 7 | 27 | **5** |
| 0007 | - | - | - | **51** |
| 0009 | 25 | 10 | 7 | **2** |
| 0010 | 13 | 8 | 17 | **4** |
| 0011 | 9 | 78 | **5** | 16 |
| 0013 | 87 | 417 | 174 | **58** |
| 0015 | **32** | 403 | 100 | 127 |
| 0017 | 24 | 44 | 85 | **8** |
| 0020 | 47 | 14 | 44 | **12** |
| 0021 | 104 | 127 | 125 | **35** |
| 0024 | 18 | 22 | 64 | **2** |
| 0027 | 49 | 44 | 19 | **3** |
| 0029 | 9 | 4 | **1** | 2 |
| 0030 | 5 | 7 | **2** | **2** |
| 0031 | 11 | 9 | **1** | 2 |
| 0035 | 28 | 20 | 10 | **7** |
| 0037 | 27 | 5 | 8 | **3** |
| 0039 | 16 | 16 | 16 | **5** |
| 0040 | 23 | 12 | 27 | **3** |
| 0043 | 11 | 26 | 5 | **2** |
| 0044 | 22 | 6 | 9 | **1** |
| 0046 | 18 | 107 | 42 | **2** |
| 0048 | 20 | 9 | 53 | **3** |
| 0049 | 20 | **2** | 16 | 5 |
| 0053 | 187 | 371 | 120 | **8** |
| 0055 | - | - | - | **180** |
| 0057 | - | - | - | **342** |
| 0058 | - | - | - | - |
| Solved Instances | 26 | 26 | 26 | 29 |

Table 3: Time (seconds) required to solve the instances in the Hanoi Tower domain. A − means that the solver (YICES or CLINGO) was not able to find a solution in 600 seconds.

seconds in all the instances, and the gap with respect to the other systems is evident also by looking at the cumulative results in Figure 4. Remarkably, in this domain ASP2IDL+SCC is also much faster than CLINGO, which solves 15 instances with solving times generally much larger.

Finally, in Table 6 there are the results on the instances of the Knight Tour with Holes domain, which is the second non-tight domain analyzed. All translation-based systems solve the same two instances (given the low number of instances solved, the plot is not reported), with ASP2IDL+SCC having an edge over ASP2IDL and LP2DIFF in terms of performance. CLINGO solves 9 instances on this domain.

The differences in performance in these two domains could be ascribed, at least partly, to the number of SCCs: indeed, while Labyrinth instances have a mean of approx. 18 SCCs, all Knight Tour instances have just one SCC. Regarding the comparison among the translation-based approaches, it has to be noted that, differently from tight programs where the reduction is fully Boolean, on non-tight domains the LP2DIFF system requires a Boolean variable as well as an integer variable for each variable in the input program.

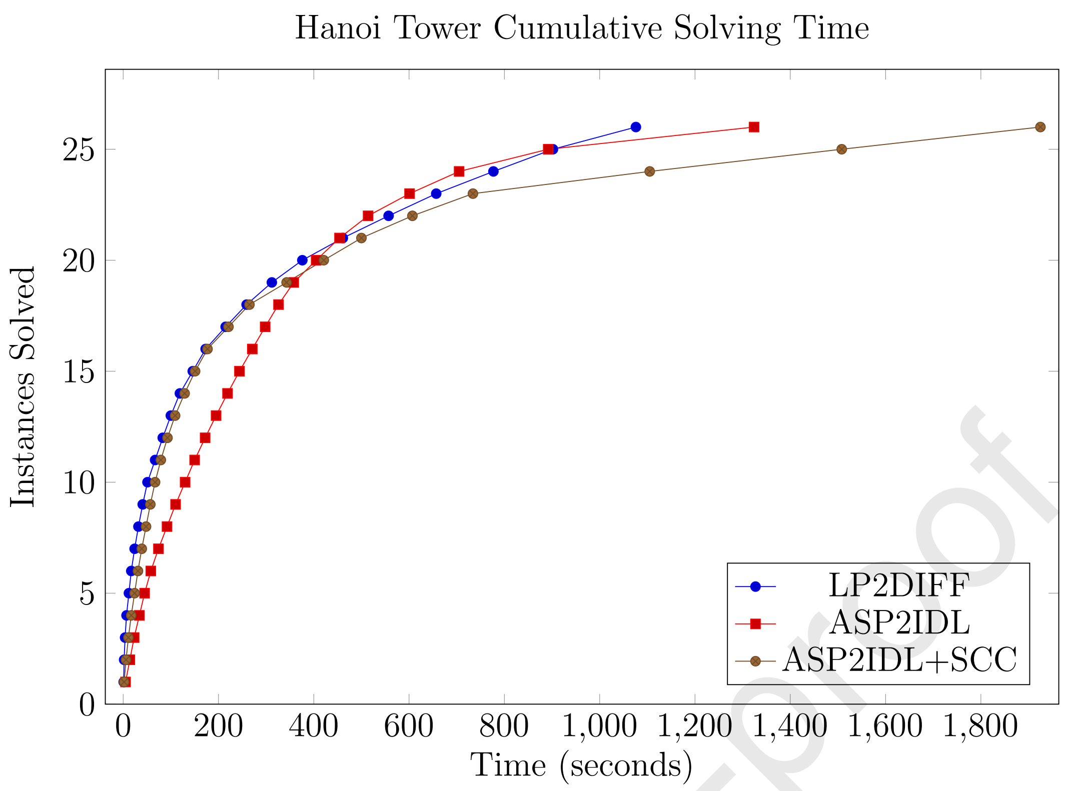


Figure 2: Cumulative time (seconds) required to solve the instances in the Hanoi domain by the ASP2IDL, ASP2IDL+SCC and LP2DIFF systems.

| Instance | ASP2IDL | ASP2IDL+SCC | LP2DIFF | CLINGO |
|---|---|---|---|---|
| 0012 | 238 | 141 | 122 | **20** |
| 0019 | - | - | - | - |
| 0020 | - | - | - | - |
| 0029 | 124 | 159 | 75 | **11** |
| 0030 | 112 | 64 | 278 | **12** |
| 0031 | 56 | 75 | 225 | **13** |
| 0037 | - | - | - | **321** |
| 0040 | - | - | - | **520** |
| 0057 | - | - | - | **530** |
| 0059 | - | - | - | **351** |
| 0060 | - | - | - | - |
| 0070 | 513 | 107 | 115 | **7** |
| 0071 | 252 | 75 | 194 | **11** |
| 0077 | - | - | - | **472** |
| 0078 | - | - | - | - |
| 0080 | - | - | - | **515** |
| 0089 | 85 | 76 | 45 | **13** |
| 0099 | - | - | - | - |
| 0100 | - | - | - | - |
| Solved Instances | 7 | 7 | 7 | 13 |

Table 4: Time (seconds) required to solve the instances in the Visit-all domain. A $-$ means that the solver (YICES or CLINGO) was not able to find a solution in 600 seconds.

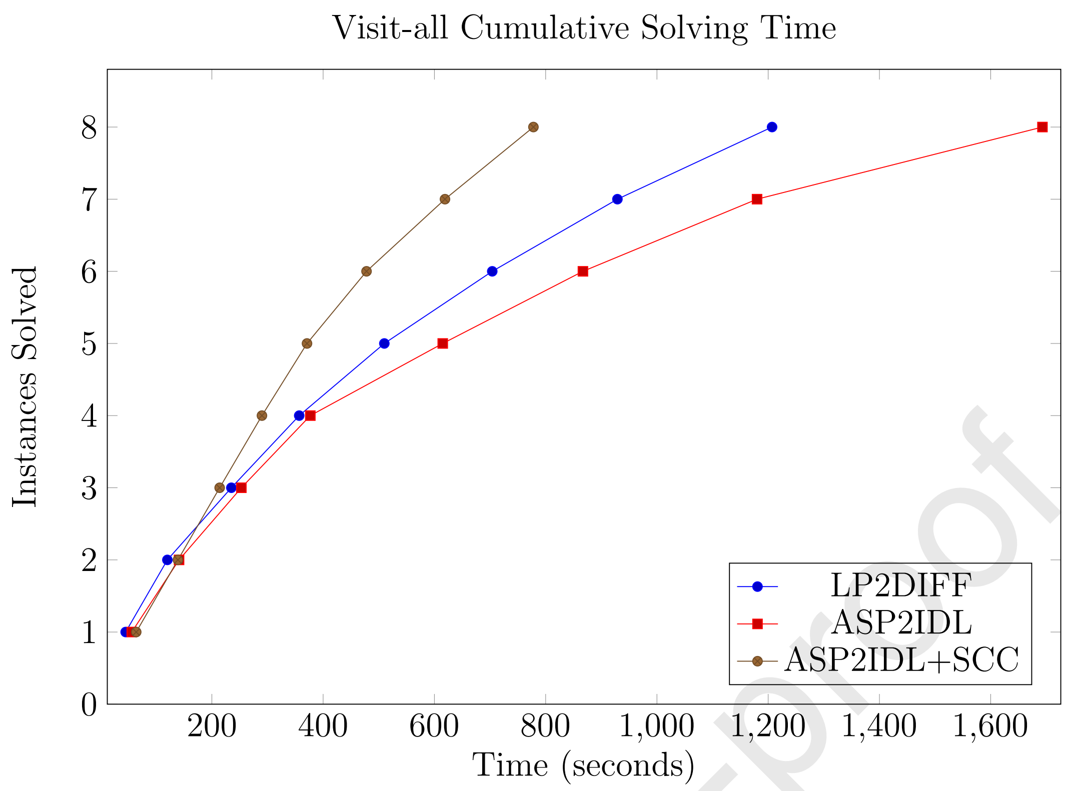


Figure 3: Cumulative time (seconds) required to solve the instances in the Visit-all domain by the ASP2IDL, ASP2IDL+SCC and LP2DIFF systems.

| Instance | ASP2IDL | ASP2IDL+SCC | LP2DIFF | CLINGO |
|---|---|---|---|---|
| 0010 | 131 | **1** | 6 | **1** |
| 0014 | - | **9** | 263 | 31 |
| 0044 | - | **9** | - | - |
| 0048 | 134 | **2** | 11 | 3 |
| 0063 | 49 | **2** | **2** | **2** |
| 0073 | 112 | **2** | 5 | 5 |
| 0092 | 49 | 4 | 18 | **2** |
| 0102 | - | **6** | 57 | 93 |
| 0124 | 103 | **2** | 77 | **2** |
| 0166 | - | **1** | - | 93 |
| 0203 | - | **6** | - | 515 |
| 0204 | - | **6** | - | - |
| 0207 | - | **6** | - | 28 |
| 0210 | - | **6** | - | - |
| 0224 | - | **7** | - | 15 |
| 0230 | - | **8** | - | - |
| 0231 | - | **8** | - | 567 |
| 0237 | - | **8** | - | 453 |
| 0240 | - | **9** | - | - |
| 0243 | - | 10 | - | **2** |
| Solved Instances | 6 | 20 | 8 | 15 |

Table 5: Time (seconds) required to solve the instances in the Labyrinth domain. A − means that the solver (YICES or CLINGO) was not able to find a solution in 600 seconds.

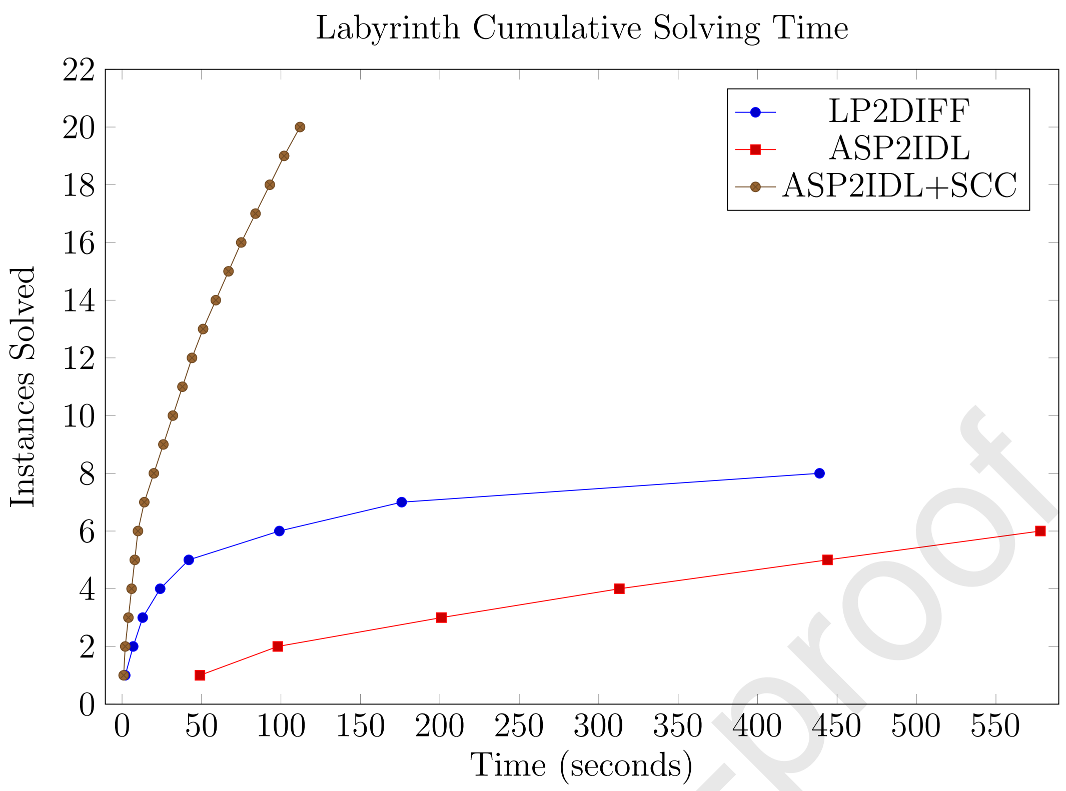


Figure 4: Cumulative time (seconds) required to solve the instances in the Labyrinth domain by the ASP2IDL, ASP2IDL+SCC and LP2DIFF systems.

| Instance | ASP2IDL | ASP2IDL+SCC | LP2DIFF | CLINGO |
|---|---|---|---|---|
| 044 | - | - | - | **7** |
| 054 | - | - | - | **2** |
| 067 | - | - | - | **3** |
| 092 | - | - | - | **4** |
| 111 | - | - | - | **4** |
| 114 | - | - | - | **4** |
| 117 | - | - | - | **12** |
| 122 | - | - | - | - |
| 130 | - | - | - | - |
| 169 | - | - | - | - |
| 213 | - | - | - | - |
| 215 | - | - | - | - |
| 216 | - | - | - | - |
| 218 | - | - | - | - |
| 227 | 8 | 3 | 5 | **2** |
| 236 | 8 | 4 | 5 | **2** |
| 240 | - | - | - | - |
| 282 | - | - | - | - |
| 289 | - | - | - | - |
| 296 | - | - | - | - |
| Solved Instances | 2 | 2 | 2 | 9 |

Table 6: Time (seconds) required to solve the instances in the Knight Tour with Holes domain. A – means that the solver (YICES or CLINGO) was not able to find a solution in 600 seconds.

## 7. Related Work

As mentioned in the introduction section, there have been several characterizations of stable models, with (possibly) corresponding reductions, and some of them are introduced throughout the paper. Here, we mention the main remaining characterizations/reductions to difference logic or other logic-based formalisms other than propositional satisfiability. Gebser et al. (2014) presented alternative target formalisms in which the acyclicity conditions can be checked using a linear representation, including difference logic (Janhunen et al., 2009). Such acyclicity conditions can be also linearly represented via SMT with Bit-Vector Logic (Nguyen et al., 2011) and Mixed Integer Programming (Liu et al., 2012). Liu and Truszczynski (2006) presented a reduction of programs with monotone and convex constraints to pseudo-Boolean constraints, based on loop formulas (Lin and Zhao, 2004; Lee and Lifschitz, 2003). The approach of Ferraris et al. (2007) is based on a syntactic transformation that turns a logic program into a formula of second-order logic similar to the formula from the definition of circumscription. Reductions have been also presented for CASP (Drescher and Walsh, 2010; Banbara et al., 2015), an extension of ASP with linear constraints (Baselice et al., 2005), but often limited to difference constraints due to their usefulness in, e.g., scheduling applications, in contrast to native approaches to handle such extension directly (e.g., Banbara et al. (2017) and the solver CLINGO[DL]). Reduction-based approaches also include those implemented in EZCSP (Balduccini and Lierler, 2017) and EZSMT (Shen and Lierler, 2018), which rely on CSP or some SMT logics (including difference logic), respectively. Possible ways to realize reasoning extensions include propagators implemented on top of ASP systems (Kaminski et al., 2023; Cuteri et al., 2020) as well as the incorporation of theories supplied by SMT solvers (Lierler, 2023), where our reduction contributes to the range of solving tools.

## 8. Conclusion

In this paper, we provide a new, simple proof-theoretic characterization of stable models and a corresponding reduction from logic programs to difference logic, which does not require Boolean variables. We implement our novel translation by means of an SMT formula and run an experimental analysis on domains from the 2017 ASP competition. Results show that our approach is competitive to and often better than LP2DIFF, and that it can also be faster than CLINGO on non-tight domains.

A current limitation of our reduction is that it considers the language fragment of the "Basic Decision" track of the ASP competition only, while the ASP-Core-2 standard (Calimeri et al., 2020) defines a more general modeling language including, e.g., aggregates, choice rules, and optimization statements. Such language extensions are already supported by translations based on acyclicity conditions (Bomanson et al., 2016), and a corresponding generalization of our reduction to map the extended language to a linear SMT encoding is a relevant topic for future work. As SMT solvers readily provide reasoning support for theories beyond difference logic, another valuable addition would be to incorporate the available logics into our translation, in order to provide means for computing stable models subject to constraints encoded in SMT.

## CRediT authorship contribution statement

**Martin Gebser:** Methodology, Formal analysis, Writing - Review & Editing; **Enrico Giunchiglia:** Conceptualization, Methodology, Formal analysis, Investigation, Writing - Original Draft, Writing - Review & Editing; **Marco Maratea:** Methodology, Investigation, Writing - Original Draft, Writing - Review & Editing; **Marco Mochi:** Methodology, Software, Validation, Investigation, Data Curation, Writing - Review & Editing, Visualization;

## Declaration of competing interest

The authors declare none.

## Data availability

A link to the source code is included in the article.

## References

Alviano, M., Amendola, G., Dodaro, C., Leone, N., Maratea, M., Ricca, F., 2019. Evaluation of disjunctive programs in WASP, in: Balduccini, M., Lierler, Y., Woltran, S. (Eds.), LPNMR, Springer. pp. 241–255.

Babovich, Y., Erdem, E., Lifschitz, V., 2000. Fages' theorem and answer set programming. CoRR cs.AI/0003042. URL: https://arxiv.org/abs/cs/0003042.

Balduccini, M., Lierler, Y., 2017. Constraint answer set solver EZCSP and why integration schemas matter. Theory and Practice of Logic Programming 17, 462–515.

Banbara, M., Gebser, M., Inoue, K., Ostrowski, M., Peano, A., Schaub, T., Soh, T., Tamura, N., Weise, M., 2015. aspartame: Solving constraint satisfaction problems with answer set programming, in: Calimeri, F., Ianni, G., Truszczyński, M. (Eds.), Proceedings of the Thirteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'15), Springer. pp. 112–126.

Banbara, M., Kaufmann, B., Ostrowski, M., Schaub, T., 2017. Clingcon: The next generation. Theory and Practice of Logic Programming 17, 408–461.

Baral, C., 2003. Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press. doi:10.1017/cbo9780511543357.

Barrett, C., Fontaine, P., Tinelli, C., 2017. The SMT-LIB Standard: Version 2.6. Technical Report. Department of Computer Science, The University of Iowa. Available at www.SMT-LIB.org.

Baselice, S., Bonatti, P.A., Gelfond, M., 2005. Towards an integration of answer set and constraint solving, in: Gabbrielli, M., Gupta, G. (Eds.), Proceedings of the 21st International Conference on Logic Programming (ICLP 2005), Springer. pp. 52–66.

Ben-Eliyahu, R., Dechter, R., 1994. Propositional semantics for disjunctive logic programs. Annals of Mathematics and Artificial Intelligence 12, 53–87.

Bomanson, J., Gebser, M., Janhunen, T., Kaufmann, B., Schaub, T., 2016. Answer set programming modulo acyclicity. Fundamenta Informaticae 147, 63–91. URL: https://doi.org/10.3233/FI-2016-1398, doi:10.3233/FI-2016-1398.

Brewka, G., Eiter, T., Truszczynski, M., 2011. Answer set programming at a glance. Communications of the ACM 54, 92–103.

Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Maratea, M., Ricca, F., Schaub, T., 2020. ASP-Core-2 input language format. Theory and Practice of Logic Programming 20, 294–309.

Calimeri, F., Gebser, M., Maratea, M., Ricca, F., 2016. Design and results of the Fifth Answer Set Programming Competition. Artificial Intelligence 231, 151–181.

Calimeri, F., Ianni, G., Ricca, F., 2014. The third open answer set programming competition. TPLP 14, 117–135.

Clark, K.L., 1978. Negation as failure, in: Logic and data bases. Springer, pp. 293–322.

Cuteri, B., Dodaro, C., Ricca, F., Schüller, P., 2020. Overcoming the grounding bottleneck due to constraints in ASP solving: Constraints become propagators, in: Bessiere, C. (Ed.), Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI 2020), ijcai.org. pp. 1688–1694.

Dodaro, C., Galatà, G., Grioni, A., Maratea, M., Mochi, M., Porro, I., 2021. An ASP-based solution to the chemotherapy treatment scheduling problem. Theory and Practice of Logic Programming 21, 835–851.

Drescher, C., Walsh, T., 2010. A translational approach to constraint answer set solving. Theory and Practice of Logic Programming 10, 465–480. doi:10.1017/S1471068410000220.

Dutertre, B., 2014a. Yices 2.2, in: Biere, A., Bloem, R. (Eds.), Proc. of the Computer Aided Verification - 26th International Conference (CAV 2014), Springer. pp. 737–744.

Dutertre, B., 2014b. Yices 2.2, in: Biere, A., Bloem, R. (Eds.), Computer Aided Verification, Springer International Publishing, Cham. pp. 737–744.

Erdem, E., Gelfond, M., Leone, N., 2016. Applications of answer set programming. AI Magazine 37, 53–68.

Erdem, E., Lifschitz, V., 2003. Tight logic programs. Theory and Practice of Logic Programming 3, 499–518.

Fages, F., 1994. Consistency of clark's completion and existence of stable models. Methods of Logic in Computer Science 1, 51–60.

Falkner, A.A., Friedrich, G., Schekotihin, K., Taupe, R., Teppan, E.C., 2018. Industrial applications of answer set programming. Künstliche Intelligenz 32, 165–176.

Ferraris, P., Lee, J., Lifschitz, V., 2007. A new perspective on stable models, in: Veloso, M.M. (Ed.), Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 372–379.

Gario, M., Micheli, A., 2015. Pysmt: A solver-agnostic library for fast prototyping of smt-based algorithms, in: SMT Workshop 2015.

Gebser, M., Janhunen, T., Rintanen, J., 2014. Answer set programming as SAT modulo acyclicity, in: Schaub, T., Friedrich, G., O'Sullivan, B. (Eds.), Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 2014), IOS Press. pp. 351–356.

Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T., 2019. Multi-shot asp solving with clingo. Theory and Practice of Logic Programming 19, 27–82. doi:10.1017/S1471068418000054.

Gebser, M., Kaufmann, B., Schaub, T., 2012. Conflict-driven answer set solving: From theory to practice. Artificial Intelligence 187, 52–89.

Gebser, M., Maratea, M., Ricca, F., 2020. The seventh answer set programming competition: Design and results. Theory and Practice of Logic Programming 20, 176–204.

Gebser, M., Obermeier, P., Schaub, T., Ratsch-Heitmann, M., Runge, M., 2018. Routing driverless transport vehicles in car assembly with answer set programming. Theory and Practice of Logic Programming 18, 520–534.

Gelfond, M., Lifschitz, V., 1988. The stable model semantics for logic programming, in: Kowalski, R.A., Bowen, K.A. (Eds.), Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, USA, August 15-19, 1988 (2 Volumes), MIT Press. pp. 1070–1080.

Gelfond, M., Lifschitz, V., 1991. Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing 9, 365–386.

Giunchiglia, E., Lierler, Y., Maratea, M., 2006. Answer set programming based on propositional satisfiability. J. Autom. Reason. 36, 345–377.

Janhunen, T., 2004. Representing normal programs with clauses, in: de Mántaras, R.L., Saitta, L. (Eds.), Proceedings of the 16th Eureopean Conference on Artificial Intelligence, (ECAI 2004), IOS Press. pp. 358–362.

Janhunen, T., Niemelä, I., Sevalnev, M., 2009. Computing stable models via reductions to difference logic, in: Erdem, E., Lin, F., Schaub, T. (Eds.), Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2009), Springer. pp. 142–154.

Kaminski, R., Romero, J., Schaub, T., Wanko, P., 2023. How to build your own asp-based system?! Theory and Practice of Logic Programming 23, 299–361. doi:10.1017/S1471068421000508.

Lee, J., Lifschitz, V., 2003. Loop formulas for disjunctive logic programs, in: Palamidessi, C. (Ed.), Proceedings of the 19th International Conference on Logic Programming (ICLP 2003), Springer. pp. 451–465.

Lierler, Y., 2023. Constraint answer set programming: Integrational and translational (or SMT-based) approaches. Theory and Practice of Logic Programming 23, 195–225. doi:10.1017/S1471068421000478.

Lifschitz, V., 2010. Thirteen definitions of a stable model, in: Blass, A., Dershowitz, N., Reisig, W. (Eds.), Fields of Logic and Computation, Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday, Springer. pp. 488–503.

Lin, F., Zhao, J., 2003. On tight logic programs and yet another translation from normal logic programs to propositional logic, in: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003), Morgan Kaufmann. pp. 853–858.

Lin, F., Zhao, Y., 2004. ASSAT: computing answer sets of a logic program by SAT solvers. Artificial Intelligence 157, 115–137.

Liu, G., Janhunen, T., Niemelä, I., 2012. Answer set programming via mixed integer programming, in: Brewka, G., Eiter, T., McIlraith, S.A. (Eds.), Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning: Proceedings (KR 2012), AAAI Press.

Liu, L., Truszczynski, M., 2006. Properties and applications of programs with monotone and convex constraints. Journal of Artificial Intelligence Research 27, 299–334.

Marek, V.W., Niemelä, I., Truszczynski, M., 2008. Logic programs with monotone abstract constraint atoms. Theory and Practice of Logic Programming 8, 167–199.

Marek, V.W., Subrahmanian, V.S., 1992. The relationship between stable, supported, default and autoepistemic semantics for general logic programs. Theoretical Computer Science 103, 365–386.

Nguyen, M., Janhunen, T., Niemelä, I., 2011. Translating answer-set programs into bit-vector logic, in: Tompits, H., Abreu, S., Oetsch, J., Pührer, J., Seipel, D., Umeda, M., Wolf, A. (Eds.), Revised Selected Papers - 19th International Conference, (INAP 2011), and 25th Workshop on Logic Programming (WLP 2011) of Applications of Declarative Programming and Knowledge Management, Springer. pp. 95–113.

Niemelä, I., 1999. Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm. Annals of Mathematics and Artificial Intelligence 25, 241–273.

Niemelä, I., 2008. Stable models and difference logic. Annals of Mathematics and Artificial Intelligence 53, 313–329.

Schüller, P., 2018. Answer set programming in linguistics. Künstliche Intelligence 32, 151–155.

Shen, D., Lierler, Y., 2018. Smt-based constraint answer set solver EZSMT+ for non-tight programs, in: Thielscher, M., Toni, F., Wolter, F. (Eds.), Proceedings of the Sixteenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2018), AAAI Press. pp. 67–71.

## Appendix

Tables 7-11 contain full details about the sizes of the generated formulas. Each table is organized as follows: the first column reports the instance, the second column contains the number of variables, while the last two columns report the number of rules of ASP2IDL and ASP2IDL+SCC. The number of variables is common to ASP2IDL and ASP2IDL+SCC.

Table 7: Number of variables and rules defined by using ASP2IDL with and without SCCs in the Graph_Coloring domain.

| Instance | # of Vars. | ASP2IDL | ASP2IDL+SCC |
|---|---|---|---|
| | | # of Rules | # of Rules |
| 0004 | 10661 | 9593 | 5783 |
| 0038 | 12503 | 11085 | 6649 |
| 0056 | 13596 | 11973 | 7173 |
| 0051 | 14096 | 12269 | 7321 |
| 0045 | 13389 | 11709 | 7001 |
| 0006 | 10593 | 9561 | 5767 |
| 0012 | 11096 | 9989 | 6021 |
| 0032 | 13104 | 11421 | 6817 |
| 0015 | 11433 | 10193 | 6123 |
| 0034 | 12717 | 11189 | 6701 |
| 0052 | 13722 | 12045 | 7209 |
| 0055 | 13677 | 12025 | 7199 |
| 0030 | 11414 | 10297 | 6215 |
| 0020 | 11173 | 10025 | 6039 |
| 0002 | 10740 | 9645 | 5809 |
| 0058 | 14120 | 12261 | 7317 |
| 0057 | 14286 | 12373 | 7373 |
| 0027 | 12151 | 10757 | 6445 |
| 0011 | 11090 | 9973 | 6013 |
| 0043 | 13531 | 11781 | 7037 |
| Mean values | 12454 | 11008 | 6600 |

Table 8: Number of variables and rules defined by using ASP2IDL with and without SCCs in the Hanoi_Tower domain.

| Instance | # of Vars. | ASP2IDL | ASP2IDL+SCC |
|---|---|---|---|
| | | # of Rules | # of Rules |
| 0007 | 239961 | 260394 | 158848 |
| 0002 | 158089 | 171926 | 104873 |
| 0035 | 166364 | 180988 | 110483 |
| 0030 | 144374 | 157198 | 95958 |
| 0055 | 157568 | 171472 | 104673 |
| 0049 | 113588 | 123892 | 75623 |
| 0013 | 139976 | 152440 | 93053 |
| 0027 | 138825 | 151110 | 92173 |
| 0043 | 135578 | 147682 | 90148 |
| 0057 | 179558 | 195262 | 119198 |
| 0031 | 148772 | 161956 | 98863 |
| 0003 | 162905 | 177130 | 108048 |
| 0029 | 139976 | 152440 | 93053 |
| 0053 | 139976 | 152440 | 93053 |
| 0037 | 139976 | 152440 | 93053 |
| 0011 | 131180 | 142924 | 87243 |
| 0017 | 122384 | 133408 | 81433 |
| 0048 | 109190 | 119134 | 72718 |
| 0020 | 135578 | 147682 | 90148 |
| 0044 | 139976 | 152440 | 93053 |
| 0040 | 153170 | 166714 | 101768 |
| 0039 | 148772 | 161956 | 98863 |
| 0058 | 183956 | 200020 | 122103 |
| 0010 | 126782 | 138166 | 84338 |
| 0024 | 124377 | 135498 | 82648 |
| 0009 | 122384 | 133408 | 81433 |
| 0021 | 139976 | 152440 | 93053 |
| 0046 | 148772 | 161956 | 98863 |
| 0001 | 153273 | 166722 | 101698 |
| 0015 | 148772 | 161956 | 98863 |
| Mean values | 146467 | 159439 | 97310 |

Table 9: Number of variables and rules defined by using ASP2IDL with and without SCCs in the Visit_all domain.

| Instance | # of Vars. | ASP2IDL # of Rules | ASP2IDL+SCC # of Rules |
|---|---|---|---|
| 0100 | 1704271 | 156705 | 123302 |
| 0029 | 157100 | 25882 | 20452 |
| 0030 | 157100 | 25882 | 20452 |
| 0019 | 927268 | 85390 | 67244 |
| 0031 | 162868 | 26494 | 20912 |
| 0060 | 1103604 | 101364 | 79786 |
| 0012 | 147678 | 24159 | 19083 |
| 0040 | 1010302 | 93064 | 73280 |
| 0071 | 200994 | 32938 | 26002 |
| 0057 | 1103604 | 101364 | 79786 |
| 0009 | 145019 | 23881 | 18874 |
| 0070 | 193746 | 32166 | 25422 |
| 0077 | 1271232 | 116808 | 91930 |
| 0099 | 1688413 | 155735 | 122574 |
| 0089 | 270711 | 44413 | 35046 |
| 0020 | 918560 | 84860 | 66846 |
| 0037 | 1000814 | 92486 | 72846 |
| 0078 | 1259404 | 116086 | 91388 |
| 0080 | 1271232 | 116808 | 91930 |
| 0059 | 1103604 | 101364 | 79786 |
| Mean values | 752263 | 77892 | 61347 |

| Instance | # of Vars. | ASP2IDL<br># of Rules | ASP2IDL+SCC<br># of Rules |
|---|---|---|---|
| 0092 | 203531 | 180128 | 138954 |
| 0231 | 460174 | 400827 | 308926 |
| 0224 | 397913 | 347288 | 267714 |
| 0204 | 341156 | 298615 | 230227 |
| 0207 | 341103 | 298593 | 230207 |
| 0237 | 460204 | 400847 | 308941 |
| 0124 | 136652 | 121960 | 94134 |
| 0044 | 529009 | 459746 | 354290 |
| 0073 | 136853 | 122026 | 94195 |
| 0243 | 529527 | 459934 | 354455 |
| 0014 | 528847 | 459654 | 354222 |
| 0010 | 86222 | 77846 | 60129 |
| 0203 | 341111 | 298573 | 230199 |
| 0240 | 460334 | 400879 | 308975 |
| 0102 | 340459 | 298341 | 229994 |
| 0063 | 167922 | 149191 | 115116 |
| 0166 | 109525 | 98261 | 75871 |
| 0048 | 136874 | 122036 | 94203 |
| 0210 | 340932 | 298515 | 230146 |
| 0230 | 459756 | 400663 | 308786 |
| Mean values | 325405 | 284696 | 219484 |

Table 10: Number of variables and rules defined by using ASP2IDL with and without SCCs in the Labyrinth domain.

Table 11: Number of variables and rules defined by using ASP2IDL with and without SCCs in the Knight Tour with Holes domain.

| Instance | # of Vars. | ASP2IDL | ASP2IDL+SCC |
|---|---|---|---|
| | | # of Rules | # of Rules |
| 044 | 257549 | 113822 | 100550 |
| 054 | 261377 | 115092 | 101670 |
| 067 | 332957 | 146170 | 129120 |
| 092 | 416229 | 181894 | 160672 |
| 111 | 509129 | 221688 | 195818 |
| 114 | 500741 | 219096 | 193531 |
| 117 | 510237 | 222010 | 196102 |
| 122 | 586153 | 257413 | 227380 |
| 130 | 593409 | 259708 | 229404 |
| 169 | 817950 | 356311 | 314725 |
| 213 | 1094145 | 473192 | 417948 |
| 215 | 1088565 | 471447 | 416409 |
| 216 | 1079193 | 468525 | 413832 |
| 218 | 1097406 | 474200 | 418837 |
| 227 | 1238553 | 535070 | 472598 |
| 236 | 1216193 | 527973 | 466339 |
| 240 | 1238317 | 535019 | 472553 |
| 282 | 1730833 | 745260 | 658232 |
| 289 | 1719265 | 741618 | 655020 |
| 296 | 1706873 | 737891 | 651733 |
| Mean values | 899753 | 390169 | 344623 |

**Declaration of interests**

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: