Agents Interoperability via Conformance Modulo Mapping

Davide Ancona, Angelo Ferrando, and Viviana Mascardi
DIBRIS, University of Genova, Italy, name.surname@dibris.unige.it

***Abstract*—We present an algorithm for establishing a flexible conformance relation between two local agent interaction protocols (LAIPs) based on mappings involving agents and messages, respectively. Conformance is in fact computed "modulo mapping": two LAIPs τ and τ' may involve different agents and use different syntax for messages, but may still be found to be conformant provided that a given map from entities appearing in τ to corresponding entities in τ' is applied. LAIPs are modelled as trace expressions whose high expressive power allows for the design of protocols that could not be specified using finite state automata or equivalent formalisms. This expressive power makes the problem of stating if τ conforms to τ' undecidable. We cope with this problem by over-approximating trace expressions that may lead to infinite computations, obtaining a sound but not complete implementation of the proposed conformance check.**

***Index Terms*—Agent Interaction Protocols, Conformance, Mappings, Trace Expressions**

I. Introduction

We open the paper by means of an example. The example allows us to explain the research question we address and to introduce the trace expressions formalism for representing agent interaction protocols in a gentle way, before their formal presentation in the body of the paper.

The example scenario is the following: the company $AI4Tour$ develops chatbots interacting with human beings in their daily working activities. $AI4Tour$ business is in the touristic sector and chatbots support touristic operators.

A typical conversation between a touristic agency $TourAgency$ and the chatbot $TravelChat$ starts with the request of whether a plane landed[1], or a cruise ship docked, or a train/bus reached the main city station; the chatbot, by accessing some database or web service in the backend, answers either "yes", "not yet", or "canc" (for canceled), and then becomes available to answer new questions.

The global agent interaction protocol (GAIP) τ which norms the simple multiagent system mas involving TA (for $TourAgency$) and TC (for $TravelChat$) might look like

$$\tau = (TA \overset{landed}{\Longrightarrow} TC{:}\epsilon \ \vee \ TA \overset{docked}{\Longrightarrow} TC{:}\epsilon \ \vee \ TA \overset{train\_arrived}{\Longrightarrow} TC{:}\epsilon \ \vee \ TA \overset{bus\_arrived}{\Longrightarrow} TC{:}\epsilon) \ \cdot \ (TC \overset{yes}{\Longrightarrow} TA{:}\epsilon \ \vee \ TC \overset{not\_yet}{\Longrightarrow} TA{:}\epsilon \ \vee \ TC \overset{canc}{\Longrightarrow} TA{:}\epsilon) \ \cdot \ \tau$$

[1]For sake of clarity, we disregard the facts that a flight is characterized by a code which should be supplied as a parameter to the query, and that when the chatbot answers and becomes ready to manage a new query, it might be able to interact with a travel agency different from $TourAgency$. The trace expressions formalism supports parameters both at the data level (to model messages which only differ for the flight code) and at the agent level (to model multiple concurrent conversations among different agents), but taking parameters into account would make the presentation more complex and we opted for keeping it as simple as possible.

where $S \overset{msg}{\Longrightarrow} R$ represents the interaction consisting of sender S sending msg to receiver R, *landed* stands for *"did the plane land?"*, *docked* stands for *"did the ship dock?"*, and so on. The : symbol represents the prefix operator between an interaction and a protocol, and ϵ represents the empty protocol. The $\vee$ symbol models exclusive choice between protocols, meaning that the travel agency can make only one request at a time among the allowed ones, and $\cdot$ represents protocol concatenation, meaning – in this example – that after receiving one request, the chatbot will react by selecting and sending one answer among the three allowed ones. Finally, $\cdot\ \tau$ means that the protocol definition is recursive: after having received and answered one question, $TravelChat$ is ready to start again.

Another company $AI4Moving$ develops chatbots that interact with citizens to provide useful information for planning a safe journey within the city boundaries.

A typical conversation between the citizen C and the chatbot $MovingChat$ starts with C asking if some ship docked (because the city traffic is highly impacted by cars and trunks disembarking), or if a train or bus just reached or will reach the main city station (because C might consider to take that bus or train, instead of the car); the chatbot answers either "yes", "in one hour", "in two hours", or "not in the next three hours", and moves to the state where it can receive new questions.

The global agent interaction protocol τ' governing mas' which involves C and MC (for $MovingChat$) is

$$\tau' = (C \overset{docked}{\Longrightarrow} MC{:}\epsilon \ \vee \ C \overset{train\_in\_station}{\Longrightarrow} MC{:}\epsilon \ \vee \ C \overset{bus\_in\_station}{\Longrightarrow} MC{:}\epsilon) \ \cdot \ (MC \overset{yes}{\Longrightarrow} C{:}\epsilon \vee MC \overset{in\_1\_h}{\Longrightarrow} C{:}\epsilon \vee MC \overset{in\_2\_h}{\Longrightarrow} C{:}\epsilon \ \vee \ MC \overset{not\_in\_3\_h}{\Longrightarrow} C{:}\epsilon) \ \cdot \ \tau'$$

When the $AI4Tour$ company acquires $AI4Moving$, it decides to keep providing the services previously offered by $AI4Moving$, but re-implementing them with its own technologies, in the most efficient and less error-prone way.

W.r.t. to the re-implementation of $MovingChat$, given that $AI4Tour$ already developed the $TravelChat$ chatbot which clearly shares some similarities with $MovingChat$, the $AI4Tour$ software engineers start wondering whether $TravelChat$ can be adapted and reused to play the role of $MovingChat$. They address the question: *"can $TravelChat$ safely substitute $MovingChat$ provided that suitable mappings between messages and between agents in mas and mas' respectively are applied?"*

The intuition behind the "mappings" the $AI4Tour$ software engineers are looking for should be clear. We formally define them as a map $M_{\mathcal{M}} : \mathcal{M}_1 \rightarrow \mathcal{M}_2$ from the messages that appear in an interaction protocol τ_{ag1} to those that appear in τ'_{ag2}, and a map $M_{\mathcal{A}} : \mathcal{A}_1 \rightarrow \mathcal{A}_2$ from the agents that appear in τ_{ag1} to those that appear in τ'_{ag2}, respectively. To answer their "substitutability" question, the engineers must:

(1) Move from the global description τ of how TA and TC interact, to TC's local agent interaction protocol τ_{TC} (LAIP):

$$\begin{aligned}\tau_{TC} = (&\xLeftarrow{landed}_{TA} : \epsilon \ \vee \ \xLeftarrow{docked}_{TA} : \epsilon \ \vee \\ &\xLeftarrow{train\_arrived}_{TA} : \epsilon \ \vee \ \xLeftarrow{bus\_arrived}_{TA} : \epsilon) \ \cdot \\ (&\xRightarrow{yes}_{TA} : \epsilon \ \vee \ \xRightarrow{not\_yet}_{TA} : \epsilon \ \vee \ \xRightarrow{canc}_{TA} : \epsilon) \ \cdot \ \tau_{TC}\end{aligned}$$

In τ_{TC} we omit to write TC as sender or receiver, as this information is implicit. Also, if there were messages in τ that involved $TravelChat$ neither as the sender not as the receiver, they would not appear in τ_{TC}.

(2) Move from the global description τ' of how citizens and MC interact, to MC's LAIP, τ'_{MC}:

$$\begin{aligned}\tau'_{MC} = (&\xLeftarrow{docked}_{C} : \epsilon \ \vee \ \xLeftarrow{train\_in\_station}_{C} : \epsilon \ \vee \\ &\xLeftarrow{bus\_in\_station}_{C} : \epsilon) \ \cdot \ (\xRightarrow{yes}_{C} : \epsilon \ \vee \ \xRightarrow{in\_1\_h}_{C} : \epsilon \ \vee \\ &\xRightarrow{in\_2\_h}_{C} : \epsilon \ \vee \ \xRightarrow{not\_in\_3\_h}_{C} : \epsilon) \ \cdot \ \tau'_{MC}\end{aligned}$$

(3) Check whether τ_{TC} is conformant to τ'_{MC}; this is achieved by looking for mappings $M_{\mathcal{A}}$ among agents and mappings $M_{\mathcal{M}}$ among messages involved in τ_{TC} and τ'_{MC}, such that $TravelChat$ can play the role of $MovingChat$ in mas', still ensuring that the GAIP τ' is respected.

(4) Select one couple of mappings among those computed in step (3), $\langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle$, based on their semantics/pragmatics.

(5) Implement a means to allow $TravelChat$ and the citizens to interact, by forcing $TravelChat$ to apply the selected mappings when interacting with them.

Agent $TourAgency$ in mas must be necessarily mapped to C in mas'. From a semantic and pragmatic point of view, the most reasonable message mapping is the one that maps $docked \in mas$ into $docked \in mas'$ (we abuse notation, and we write $msg \in mas$ to mean that msg is one of the messages exchanged by agents belonging to mas); $train\_arrived$ into $train\_in\_station$; $bus\_arrived$ into $bus\_in\_station$; $yes \in mas$ into $yes \in mas'$; not_yet into in_2_h; and $canc$ into not_3_h. The $landed$ message is mapped into no message: when "pretending to be $MovingChat$", $TravelChat$ will never receive a message whose meaning is close to $landed$, as τ' does not support it. On the other hand, $TravelChat$ is not able to discriminate between trains and buses arriving in one or two hours. The mapping of not_yet into in_2_h is a cautious choice and the citizen will never receive the message in_1_h, even if it would be supported by τ'.

From a purely syntactic point of view, and considering protocol specifications only – hence, disregarding the actual services and actions that are triggered by reception of messages –, many other mappings would respect the protocol conformance, including the one that maps $canc$ into $yes \in mas'$ and $yes \in mas$ into not_3_h.

The research question that we address in this paper is the one in step (3) above. We point out that such research question cannot be answered by using ontology matching algorithms [1]. Ontology matching techniques could indeed be exploited in step (4) of the process, as we discuss in the Conclusions, but not in step (3): an ontology represents static knowledge, not dynamic behaviour. An agent interaction protocol represents dynamic behaviour, not static knowledge. Checking whether a protocol is conformant to another must necessarily take such dynamics into account, which is not required in an ontology matching process and which raises many subtle issues. For example, when moving from τ to τ' to substitute ag', ag must be capable to react *at least* to all the "passive events" (for example, receiving a message) that ag' can address, and to perform *at most* all the "active events" (for example, sending a message) that ag' can perform, at any stage of the protocol. This requirement cannot be satisfied by an ontology matching approach, where it does not even make sense, whereas it is well known in the protocol conformance literature. Depending on the expressiveness of the language used to specify GAIPs, verifying that ag can actually substitute ag' in a safe way may be more or less complex, or even impossible to perform in an exact way. As an example, recursive protocol definitions are usually disregarded in the literature as they are extremely complex to manage. The formalism we use for modeling GAIPs and LAIPs supports recursion, and this is enough to make existing conformance checking algorithms not powerful enough for our needs.

Our contribution is an algorithm for addressing step (3) above when GAIPs are specified as trace expressions [2], [3], [4], [5], [6], [7], [8], [9]. To demonstrate the feasibility of our approach, we present an example implemented in JADE [10].

II. Related Work

The works closer to our proposal come from Baldoni and Baroglio who, together with their colleagues, introduced the notion of syntactic conformance in the context of interaction protocols for MAS and Service Oriented Computing (SOC) scenarios, starting from 2004. Conformance is based on the notion of interoperability among the entities' policies (e.g. a BPEL process [11], similar to some extent to our LAIPs) with respect to interaction protocols (e.g. a WS-CDL choreography [12], similar to our GAIPs), through the use of finite state automata. While in [13], [14], [15] protocols were limited to involve two entities only, [16] presents an extension supporting multiple parties. A further extension is presented in [17] where decision points are explicitly represented.

Besides the fact that we address the conformance between LAIPs, there are other differences between those works and ours: first, they assume that entities/messages involved in the policy and in the protocol respectively, are exactly the same in order for the conformance check to have some chance to succeed: no notion of mapping is foreseen; second, the expressive power of trace expressions is higher than the expressive power of WS-CDL/BPEL. The presence of expansive subtraces, introduced later on, makes trace expressions able to recognize context-free and non context-free languages, and raises technical problems that do not show up when less expressive formalisms are used.

Among the works by Baldoni and Baroglio's team, however, the most inspiring for our research is [18], recently improved and extended in [19]. That work presents an agent *typing system*, where types are defined as commitments [20]. The typing includes a notion of compatibility, based on subtyping, which allows for the safe substitution of agents to roles along an interaction that is ruled by a commitment-based protocol. The proposal is implemented in the 2COMM framework [21] which is based on the Agent & Artifact meta-model [22], and exploits JADE and CArtAgO [23]. Considering the LAIP associated with an agent as its "communicative type" is an almost natural idea in our approach also. The LAIP makes the communicative interface of an agent explicit and can be used both to type check an agent w.r.t. the possibility of entering a MAS normed by some GAIP, and to define a subtyping relation which we name "is conformant to" relation. The main difference between our approach and the one discussed in [18] lies in the adopted formalism and the generality: commitments without mappings there, trace expressions with mappings here.

Many other works besides those mentioned above aim at defining and testing conformance in the SOC community, including [24], [25], [26], [27]. None of them uses formalisms which are as powerful as context-free grammars, or more, and none integrates the notion of agents and messages mappings. Also, some of them are limited to two-party protocols.

When moving to the MAS realm, we can devise the same differences between our approach and the others as those identified for SOC approaches: lower expressive power of the adopted formalisms and less generality, due to the absence of mappings in the conformance definition. Among the most notable contributions to protocol conformance, we may mention [28] where Endriss *et al.* identify three levels of conformance, weak, exhaustive, and robust, and explore how a specific class of logic-based agents can exploit an AIP formalism based on simple if-then rules to check conformance a priori or enforce it at runtime. In a similar way, Alberti *et al.* exploit the *S*CIFF abductive proof-procedure [29] for both a priori and runtime verification of compliance of agent interactions [30]. In [31], Chopra and Singh formalize the notions of conformance, coverage and interoperability. In [32] a formal interoperability test for agents is presented. That work considers the presence of two agents only, but in an open scenario where agents can behave differently from the protocol specification. Finally, in [33], Giordano and Martelli address the problem of conformance between an agent and a protocol through an automata-based technique, when the specification of the protocol is given in a temporal action logic.

III. Background and Basic Definitions

a) Trace expressions: Trace expressions are based on events and event types and can be combined with various operators. For sake of presentation, in this paper we do not distinguish between events and event types, and we trace the last ones back to the notion of *interactions*.

An interaction is represented by $S \overset{msg}{\Longrightarrow} R$ where S is the sender, msg is the message, and R is the receiver. We define MSG as the function which, given an interaction, returns its message: $MSG(S \overset{msg}{\Longrightarrow} R) = msg$.

A trace expression τ represents a set of possibly infinite interaction traces and is defined on top of the following operators:
– ϵ (empty trace), denoting the singleton set $\{\epsilon\}$ containing the empty interaction trace ϵ,
– $int{:}\tau$ (*prefix*), denoting the set of all traces whose first interaction is int and the remainder is a trace of τ,
– $\tau_1 {\cdot} \tau_2$ (*concatenation*), denoting the set of all traces obtained by concatenating the traces of τ_1 with those of τ_2,
– $\tau_1 \wedge \tau_2$ (*intersection*), denoting the intersection of the traces of τ_1 and τ_2,
– $\tau_1 \vee \tau_2$ (*union*), denoting the union of the traces of τ_1 and τ_2,
– $\tau_1 | \tau_2$ (*shuffle*), denoting the set obtained by shuffling the traces in τ_1 with the traces in τ_2.

To support recursion without introducing an explicit construct, trace expressions are regular terms and can be represented by a finite set of syntactic equations.

As an example, $T = int{:}T$ is equivalent to the infinite but regular term $int{:}int{:}int{:}int{:}\ldots$. The only trace represented by T is int^{ω}: trace expressions are interpreted in a coinductive way to represent infinite traces of interactions [34].

The semantics of trace expressions is specified by a transition relation $\delta \subseteq \mathcal{T} \times \mathcal{I} \times \mathcal{T}$, where $\mathcal{T}$ and $\mathcal{I}$ denote the set of trace expressions and of interactions, respectively. Notation $\tau_1 \xrightarrow{int} \tau_2$ means $(\tau_1, int, \tau_2) \in \delta$; the transition $\tau_1 \xrightarrow{int} \tau_2$ expresses the property that the system can safely move from the state specified by τ_1 into the state specified by τ_2 when interaction int takes place. **Trace expressions model GAIPs**.

b) Expansive trace expressions: The expressive power of trace expressions is due to the presence of expansive terms.

Def. 3.1: A trace expression τ is expansive iff $\tau = \tau_1 {\cdot} \tau_2$ and τ_1 is a cyclic term containing τ; or $\tau = \tau_1 | \tau_2$ and either τ_1 or τ_2 is a cyclic term containing τ; or $\tau = \tau_1 \wedge \tau_2$ and either τ_1 or τ_2 is a cyclic term containing τ; or it contains a subtrace that is expansive.

Expansive subtraces allow the trace expression formalism to recognize more than context-free languages. Given a trace expression τ, $exp(\tau)$ is true if τ is expansive.

c) Trace expression over-approximation: Given a trace expression τ, $\widetilde{\tau}$ is an over-approximation of τ iff τ is not expansive and $\widetilde{\tau} = \tau$; or τ is expansive and $\widetilde{\tau}$ is a trace expression equivalent to a regular expression representing a superset of the traces recognized by τ.

Since $\widetilde{\tau}$ is equivalent to a regular expression, it is not expansive. Given an expansive trace expression τ, there may be many $\widetilde{\tau}$ that over-approximate it. The algorithm that computes one of these non-expansive over-approximations is discussed in [35].

d) Projection: Let $\mathcal{A}$ be a set of agents. Projection [36] is a function $\Pi : \mathcal{T} \times \mathcal{P}(\mathcal{A}) \rightarrow \mathcal{T}$. Given a trace expression τ and a set of agents $ags \subseteq \mathcal{A}$ as input, Π returns a trace expression τ_{ags} which contains only interactions involving agents in ags: interactions that do not involve agents in ags are removed from τ_{ags}. Since in this paper we are interested in projecting onto a single agent at a time, we will write τ_{ag} instead of $\tau_{\{ag\}}$ to denote the projection of τ onto agent ag.

When projected onto S, interaction $S \overset{msg}{\Longrightarrow} R$ is represented by $\overset{msg}{\Longrightarrow}_R$ ("sending interaction"); when projected onto R, it is represented by $\overset{msg}{\Longleftarrow}_S$ ("receiving interaction"). In projected interactions, we omit to write the agent onto which the projection is performed. We extend the MSG function introduced for interactions, to sending and receiving interactions: $MSG(\overset{msg}{\Longrightarrow}_R) = MSG(\overset{msg}{\Longleftarrow}_R) = msg$. **Projected trace expressions model LAIPs**.

e) GAIPs, agents, interactions, and messages: Let mas be a multiagent system governed by some GAIP modeled by trace expression τ. We define $GAIP(mas)$ as τ. Let τ be a trace expression involving all and only agents $\mathcal{A}$ and interactions $\mathcal{I}$. We define $AG(\tau)$ as $\mathcal{A}$, $INT(\tau)$ as $\mathcal{I}$, and $MSG(\tau)$ as $\{msg \mid int \in INT(\tau) \text{ and } msg = MSG(int)\}$.

The definitions of AG, INT and MSG hold for both trace expressions and projected trace expressions.

IV. LAIP Conformance Modulo Mapping

We first give a simpler, but stronger, definition of compliance which does not allow renaming of messages and agents.

Def. 4.1: Given two LAIPs τ_{ag1} and τ'_{ag2}, we say that τ_{ag1} is conformant to τ'_{ag2}, written $\tau_{ag1} \leq \tau'_{ag2}$, iff the following conditions are coinductively verified:

- $\forall msg, ag$ if $\exists \tau''_{ag1}$ s.t. $\tau_{ag1} \xrightarrow{\overset{msg}{\Longrightarrow}_{ag}} \tau''_{ag1}$, then $\exists \tau'''_{ag2}$ s.t. $\tau'_{ag2} \xrightarrow{\overset{msg}{\Longrightarrow}_{ag}} \tau'''_{ag2} \wedge \tau''_{ag1} \leq \tau'''_{ag2}$;
- $\forall msg, ag$ if $\exists \tau'''_{ag2}$ s.t. $\tau'_{ag2} \xrightarrow{\overset{msg}{\Longleftarrow}_{ag}} \tau'''_{ag2}$, then $\exists \tau''_{ag1}$ s.t. $\tau_{ag1} \xrightarrow{\overset{msg}{\Longleftarrow}_{ag}} \tau''_{ag1} \wedge \tau''_{ag1} \leq \tau'''_{ag2}$;
- $\{\tau'''_{ag2} \mid \exists msg, ag. \tau'_{ag2} \xrightarrow{\overset{msg}{\Longrightarrow}_{ag}} \tau'''_{ag2}\} \neq \emptyset$ implies $\{\tau''_{ag1} \mid \exists msg, ag. \tau_{ag1} \xrightarrow{\overset{msg}{\Longrightarrow}_{ag}} \tau''_{ag1}\} \neq \emptyset$.

In the following formalization we assume that $ag1$ is an agent in mas, and $ag2$ an agent in mas', and define $\tau = GAIP(mas)$, $\tau' = GAIP(mas')$, $\tau_{ag1} = \Pi(\tau, ag1)$, $\tau'_{ag2} = \Pi(\tau', ag2)$, $\mathcal{A}_1 = AG(\tau_{ag1})$, $\mathcal{A}_2 = AG(\tau'_{ag2})$, $\mathcal{M}_1 = MSG(\tau_{ag1})$, $\mathcal{M}_2 = MSG(\tau'_{ag2})$.

As introduced in Section I, we consider a map $M_{\mathcal{M}} : \mathcal{M}_1 \rightarrow \mathcal{M}_2$ from the messages that appear in τ_{ag1} to those that appear in τ'_{ag2}, and a map $M_{\mathcal{A}} : \mathcal{A}_1 \rightarrow \mathcal{A}_2$ from the agents that appear in τ_{ag1} to those that appear in τ'_{ag2}.

A more general conformance relation modulo mappings can be defined in terms of the basic conformance relation of Definition 4.1.

Def. 4.2: Given two LAIPs τ_{ag1} and τ'_{ag2}, and two mappings $M_{\mathcal{M}}$ and $M_{\mathcal{A}}$ on messages and agents, respectively, we say that τ_{ag1} is conformant to τ'_{ag2} modulo $M_{\mathcal{M}}$ and $M_{\mathcal{A}}$, written $\tau_{ag1} \leq_{\langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle} \tau'_{ag2}$, iff $\langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle (\tau_{ag1}) \leq \tau'_{ag2}$.

With $\langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle (\tau_{ag1})$ we denote the trace expression obtained from τ_{ag} by replacing all the interactions $\overset{msg}{\Longrightarrow}_{ag}$ and $\overset{msg}{\Longleftarrow}_{ag}$ with $\overset{M_{\mathcal{M}}(msg)}{\Longrightarrow}_{M_{\mathcal{A}}(ag)}$ and $\overset{M_{\mathcal{M}}(msg)}{\Longleftarrow}_{M_{\mathcal{A}}(ag)}$, respectively.

Intuitively, the relation $\tau_{ag1} \leq_{\langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle} \tau'_{ag2}$ ensures that $ag1$ can safely substitute $ag2$ in mas', provided that mappings $M_{\mathcal{M}}$ and $M_{\mathcal{A}}$ are applied to messages and agents in τ_{ag1}, respectively.

An algorithm for conformance. Given the definitions 4.1 and 4.2, a first question that may arise is whether there exists an algorithm for deciding if the compliance relation holds for a pair of trace expressions, and, in case of the more general notion of conformance modulo mappings, if such mappings can be computed. Unfortunately, the problem is undecidable even for the simpler conformance relation of Definition 4.1; this can be derived by the fact that a context-free grammar can be encoded into a trace expression, and that the problem of inclusion between context-free languages (which is known to be undecidable) can be reduced to the conformance problem between two trace expressions. Despite this negative result, it is still interesting to investigate the existence of algorithms which are sound (even though not complete) w.r.t. the definition of conformance between trace expressions.

We define the merging of two maps in the following way: let $M_{\mathcal{M}} : \mathcal{M}1 \rightarrow \mathcal{M}2$ and $M'_{\mathcal{M}} : \mathcal{M}1' \rightarrow \mathcal{M}2'$ be two maps among messages:

if $\exists_{msg \in \mathcal{M}1 \cap \mathcal{M}1'} . M_{\mathcal{M}}(msg) \neq M'_{\mathcal{M}}(msg)$
then $merge(M_{\mathcal{M}}, M'_{\mathcal{M}}) = \emptyset$
else $merge(M_{\mathcal{M}}, M'_{\mathcal{M}}) = M''_{\mathcal{M}} : \mathcal{M}1 \cup \mathcal{M}1' \rightarrow \mathcal{M}2 \cup \mathcal{M}2'$ such that $M''_{\mathcal{M}} = M_{\mathcal{M}} \cup M'_{\mathcal{M}}$.

In other words, merging two maps consists in computing the union of the elements in the maps, unless there is some conflict, namely, some element is mapped to two different elements in the two maps. In this case, the maps cannot be merged (the merged map is empty). For instance, if $M_{\mathcal{M}} = \{msg_1 \mapsto msg_2, msg_3 \mapsto msg_4\}$ and $M'_{\mathcal{M}} = \{msg_3 \mapsto msg_4, msg_5 \mapsto msg_6\}$, the merged map is $M''_{\mathcal{M}} = \{msg_1 \mapsto msg_2, msg_3 \mapsto msg_4, msg_5 \mapsto msg_6\}$. If $M_{\mathcal{M}} = \{msg_1 \mapsto msg_2\}$ and $M'_{\mathcal{M}} = \{msg_1 \mapsto msg_3\}$, their merged map is empty.

The same definition can be adopted for merging maps of agents.

Given two maps $M_{\mathcal{M}}$ and $M_{\mathcal{A}}$, a sending interaction $\overset{msg}{\Longrightarrow}_R$ can substitute a sending interaction $\overset{msg'}{\Longrightarrow}_{R'}$ in the context of $M_{\mathcal{M}}$ and $M_{\mathcal{A}}$ iff $merge(\{msg \mapsto msg'\}, M_{\mathcal{M}}) \neq \emptyset$ and $merge(\{R \mapsto R'\}, M_{\mathcal{A}}) \neq \emptyset$. The definition for a receiving interaction $\overset{msg}{\Longleftarrow}_S$ substituting a receiving interaction $\overset{msg'}{\Longleftarrow}_{S'}$ is similar.

The computation of $\tau_{ag1} \leq_{\langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle} \tau'_{ag2}$ is carried out by the “isConformant” algorithm. The algorithm starts from two initial agent and message maps, and incrementally adds to them those mappings which are necessary to ensure agents interoperability. Consequently, it is possible to obtain partial maps where some messages and agents have not been mapped to anything at the end of the computation. Partial maps must be completed (namely, they must become total maps and be defined on all the elements in their domain), in order to be used in practice. Completion can be achieved by adding dummy elements in the range, and associate the elements in the domain that had no corresponding element in the range, with such dummy elements. The completion step is necessary to ensure that, when actually used to substitute $ag1 \in mas$ to $ag2 \in mas'$, the maps returned by the algorithm can be applied to all the agents and messages appearing in τ_{ag1}. The isConformant algorithm operates by cases, and the following implication holds:

$$\tau_{ag1} \leq_{\langle cM_{\mathcal{M}}, cM_{\mathcal{A}} \rangle} \tau'_{ag2} \impliedby$$

$\langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle$ = isConformant(τ_{ag1}, τ'_{ag2}, $\emptyset$, $\{ag1 \mapsto ag2\}$) **and** $\langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle \neq \langle \emptyset, \emptyset \rangle$ **and** $complete(M_{\mathcal{M}}) = cM_{\mathcal{M}}$ **and** $complete(M_{\mathcal{A}}) = cM_{\mathcal{A}}$, where $complete$ is the map completion step sketched above.
Conformance can be lifted to global protocols:

$$\tau \leq \tau' \iff \forall_{ag_i \in mas} . \exists_{ag'_j \in mas'} . \tau_{ag_i} \leq \tau'_{ag'_j}$$

$$\tau \leq_{\langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle} \tau' \iff \forall_{ag_i \in mas} . \exists_{ag'_j \in mas'} . \tau_{ag_i} \leq_{\langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle} \tau'_{ag'_j}$$

Conjecture: Soundness. The conformance algorithm is sound w.r.t. Definition 4.2.
Claim: No Completeness. The algorithmic implementation of the conformance test is not complete.

For space constraints, the pseudo-code of isConformant along with the sketch of the claim proof is available at https://www.disi.unige.it/person/MascardiV/Download/supplementalConformanceModuloMapping.pdf.

V. Examples and Implementation

Given two MASs mas and mas' ruled by $\tau = GAIP(mas)$ and $\tau' = GAIP(mas')$ respectively, the steps for using an agent involved in mas inside mas' are the following: (i) identify the agent $ag1 \in mas$ to be used in mas'; (ii) generate the set of agents and maps that ensure τ_{ag1} conformance $\{(ag2, \langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle) | ag2 \in mas', \tau_{ag1} \leq_{\langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle} \tau'_{ag2}\}$; (iii) select one couple of agents and maps $(ag2, \langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle)$ from the set, based on domain-dependent criteria that might involve message similarity, similar behaviours of the mapped agents, and so on; (iv) generate an interface i for $ag1$ driven by $(ag2, \langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle)$; (v) substitute $ag2$ in mas' with the “interfaced version” of $ag1$: the agents in the MAS obtained via this substitution still respect τ', because of step (ii).

The notion of *interface* introduced in step (iv) is related to the actual use of the maps generated during the conformance check and it is meant as a logical component offering a “bridging service”, that can be implemented in many different ways. Hence, the interface i generated in step (iv) realizes the map-driven translations needed by agents in mas and mas' to interoperate. Each time $ag1 \in mas$ performs the action of sending a message msg_1 to an agent $ag1_r \in mas$, the interface i “intercepts” (from a logical point of view) msg_1, translates it into $M_{\mathcal{M}}(msg_1) = msg_2$, and forwards msg_2 to $M_{\mathcal{A}}(ag1_r) \in mas'$. In the same way, each time an agent $ag2_s \in mas'$ sends a message msg_3 to $ag2 \in mas'$, the interface intercepts msg_3, looks for a message msg_4 that $ag1$ can receive in the current protocol state s.t. $M_{\mathcal{M}}(msg_4) = msg_3$, translates msg_3 into msg_4, and forwards it to $ag1$.

As an example, let us consider a GAIP τ representing the protocol where an agent $Buyer$ (Buy) asks to an agent $Seller$ (Sel) for a resource and if the resource is available, the $Seller$ can give it in exchange of money, otherwise the $Seller$ informs the $Buyer$ of the unavailability.

$$\tau = (Buy \overset{res?}{\Longrightarrow} Sel):$$

$$(Sel \overset{res}{\Longrightarrow} Buy{:}Buy \overset{money}{\Longrightarrow} Sel{:}\epsilon \vee Sel \overset{no}{\Longrightarrow} Buy{:}\tau)$$

Let us consider another similar GAIP τ' defining a bookshop protocol where an agent $Client$ (Cl) asks for a book to an agent $BookShop$, and again if the book is available the $BookShop$ ($Shop$) agent sells it for a given amount of euros, otherwise a no_avbl message is returned.

$$\tau' = (Cl \overset{book?}{\Longrightarrow} Shop):$$

$$((Shop \overset{book}{\Longrightarrow} Cl{:}Cl \overset{euros}{\Longrightarrow} Shop{:}\epsilon) \vee (Shop \overset{no\_avbl}{\Longrightarrow} Cl{:}\tau'))$$

We want to check if τ is conformant to τ', $\tau \leq \tau'$. From the definition of conformance between global protocols, for each agent $ag \in \tau$ we must find at least one agent $ag' \in \tau'$ s.t. $\tau_{ag} \leq_{\langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle} \tau'_{ag'}$. First of all, we generate the local perspectives LAIPs of τ and τ' through projection.

$$\tau_{Buy} = (\overset{res?}{\Longrightarrow}_{Sel}){:}((\overset{res}{\Longleftarrow}_{Sel} : \overset{money}{\Longrightarrow}_{Sel} {:}\epsilon) \vee (\overset{no}{\Longrightarrow}_{Sel} {:}\tau_{Buy}))$$

$$\tau_{Sel} = (\overset{res?}{\Longleftarrow}_{Buy}){:}((\overset{res}{\Longrightarrow}_{Buy} : \overset{money}{\Longleftarrow}_{Buy} {:}\epsilon) \vee (\overset{no}{\Longleftarrow}_{Buy} {:}\tau_{Sel}))$$

$$\tau'_{Cl} = (\overset{book?}{\Longrightarrow}_{Shop}){:}((\overset{book}{\Longleftarrow}_{Shop} : \overset{euros}{\Longrightarrow}_{Shop} {:}\epsilon) \vee (\overset{no\_avbl}{\Longleftarrow}_{Shop} {:}\tau'_{Cl}))$$

$$\tau'_{Shop} = (\overset{book?}{\Longleftarrow}_{Cl}){:}((\overset{book}{\Longrightarrow}_{Cl} : \overset{euros}{\Longleftarrow}_{Cl} {:}\epsilon) \vee (\overset{no\_avbl}{\Longrightarrow}_{Cl} {:}\tau'_{Shop}))$$

Then we apply the rules deriving from Definition 4.2, obtaining that (Figure 1c) $\tau_{Buy} \leq_{\langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle} \tau'_{Cl}$ with $M_{\mathcal{M}} = \{res? \mapsto book?,\ res \mapsto book,\ money \mapsto euros,\ no \mapsto no\_avbl\}$ and $M_{\mathcal{A}} = \{Buy \mapsto Cl,\ Sel \mapsto Shop\}$ and (Figure 1d) $\tau_{Sel} \leq_{\langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle} \tau'_{Shop}$ with the same maps. From this, we derive that $\tau \leq_{\langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle} \tau'$.

In this example we had no prior knowledge on possibly correct mappings. In many real scenarios, however, some of the correct mappings among messages and agents, respectively, are known in advance. The values $\emptyset$, $\{ag1 \mapsto ag2\}$ used to initialize the maps in the definition of $\tau_{ag1} \leq_{\langle cM_{\mathcal{M}}, cM_{\mathcal{A}} \rangle} \tau'_{ag2}$ **iff** $\langle M_{\mathcal{M}}, M_{\mathcal{A}} \rangle$ = isConformant(τ_{ag1}, τ'_{ag2}, $\emptyset$, $\{ag1 \mapsto ag2\}$) correspond to the worst case where the developer has no knowledge at all about the possible correct mappings, and wants to generate all of them for further inspection. If the developer knows that the initial associations modelled by

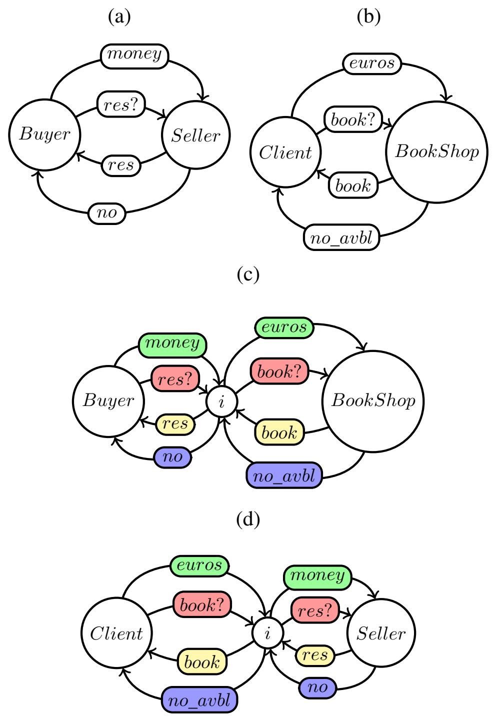

Fig. 1: (a) and (b) MASs presented in the example; (c) *Buyer* substitutes *Client* through the interface i driven by $M_{\mathcal{A}} = \{Buyer \mapsto Client, Seller \mapsto BookShop\}$, $M_{\mathcal{M}} = \{res? \mapsto book?, res \mapsto book, money \mapsto ack, no \mapsto no\_avbl\}$); (d) *Seller* substitutes *BookShop* with an interface driven by the same maps.

$M_{\mathcal{M}_0}$ and $M_{\mathcal{A}_0}$ must hold, he/she can run isConformant(τ_{ag1}, τ'_{ag2}, $M_{\mathcal{M}_0}$, $M_{\mathcal{A}_0}$), forcing the algorithm to extend such initial knowledge with new associations or to answer that, with these maps, the protocols are not conformant. This would be the case for example in Software Product Line applications and in evolving IoT scenarios where the developer knows which components in the previous product version/system should be replaced by which in the new one, and wants to check if it is possible, respecting the existing communication protocols.

Although introducing the notion of interface for showing how we can use the mappings generated during the conformance check (step (5) presented in Section I deals with exploiting mappings in practice) makes the presentation almost simple and intuitive, the actual implementation of such an interface requires to take care of many aspects dependent on the adopted MAS framework. Step (5) is in fact heavily application-dependent and can be faced in different ways, depending on the applications constraints: it would be possible for example to insert a "translator agent" into mas', that intercepts and manages all interactions involving ag; or to create a wrapper for agent ag that allows it to automatically translate incoming and outgoing messages according to $M_{\mathcal{M}}$; or even to automatically modify ag's source code, if available, to hard-wire the $M_{\mathcal{M}}$ mapping at source code level. Whatever the approach is, all the agents in mas' should be aware that ag' has been substituted by ag, in order to handle communication properly.

To demonstrate how to exploit the maps generated by the conformance testing to substitute JADE (http://jade.tilab.com) agents with other JADE agents, we adopted an automatic source code translation approach. The methodology we followed consists in three steps:

a) 1st step, conformance checking (corresponding to steps (i),(ii) and (iii) presented at the very beginning of this Section): The algorithm presented in Section IV is fully implemented in SWI-Prolog (http://www.swi-prolog.org). Prolog has been chosen thanks to its built-in support to cyclic terms, coinduction, and for the possibility to use backtracking for generating all the existing maps. The implementation of the algorithm is $<$ 400 LOC.

b) 2nd step, substitution (corresponding to step (iv)): Let us suppose that $ag1 \in mas$ can substitute $ag2 \in mas'$ with maps $M_{\mathcal{A}}$ and $M_{\mathcal{M}}$. For demonstrating how substitution can be put into practice we have opted for a basic approach where we apply the maps to the source code of $ag1$, and we use the modified source code $map(ag1)$ instead of the source code of $ag2$ in mas': we operate on the Java file containing the JADE class implementing $ag1$ and we substitute all the occurrences of ag_i with $M_{\mathcal{A}}(ag_i)$ and all occurrences of msg_i with $M_{\mathcal{M}}(msg_i)$. This substitution step is also implemented in SWI-Prolog ($<$ 30 LOC).

c) 3rd step, execution (corresponding to step (v)): We recompile $map(ag1)$ and we execute mas' with $map(ag1)$ instead of $ag2$. Despite being simple and applicable only when the source code is available, this approach demonstrates how we can actually use the maps generated during the conformance check, and has been adopted for all the examples shown in this paper.

VI. Conclusions and Future Work

In this paper we have presented a conformance modulo mapping algorithm suitable for checking conformance between local protocols specified as (projected) trace expressions, together with its implementation and usage example. The paper presents a general solution to the problem with as few constraints as possible, to make it reusable in as many situations as possible, but the actual scenarios where we believe that our approach can be more profitably exploited involve conformance between different versions of the same LAIP or LAIPs which are known to be similar, like the ones presented in Sections I and V. As another example, a self-driving car may interact with other cars, lights, etc., according to the current road norms (LAIP τ_1). Norms change and the new LAIP to which the car must conform, becomes τ_2. Which transformations (mappings) should we implement over τ_1 to ensure it is syntactically conformant to τ_2? The developer in charge of migrating τ_1 to τ_2 can use our algorithm for having guarantees on the syntactic compliance, although he/she cannot have guarantees that semantics is preserved: a human is required to finally select and validate the produced mappings. We think that semantic compliance will never be fully automatized, and for this reason we expect that our algorithm should be used in scenarios where LAIPs should not be re-aligned frequently.

W.r.t. the five steps introduced in Section I, we exploited achieved results for steps (1-2), we devoted the entire paper to step (3), and we demonstrated how to tackle step (5). More sophisticated approaches could be followed for step (5), each with pros and cons. Experimenting some of them, such as introducing a mediator agent between mas and mas' acting as the i interface and generating wrappers for the agents that must substitute other agents, will be explored in the future.

Step (4) is an open problem which falls outside the scope of our investigation: in this paper we do not face the issue of "semantic/pragmatic conformance", but only that of "syntactic conformance". In case some constraints on interactions are know, for example commitments that must be fulfilled and that can drive the choice of the most suitable mapping, we might exploit them. Otherwise, by interpreting messages as words or sentences in some natural language, we might take advantage of semantic techniques similar to those used for matching ontological concepts [37]. Ontology matching could hence be exploited in the global process we have presented, in step (4). We remark that if we knew in advance which are the semantically correct message and agents mappings, we could feed our algorithm with them and use it as a "plain conformance checker" like those mentioned above, rather than a "conformant mappings builder". Even if we knew all the correct mappings in advance, however, we could not run any of the existing conformance checking algorithms on trace expressions, because of their higher expressiveness.

Finally, an extremely challenging issue would be to identify mappings where one message used in one MAS corresponds to a *sequence* of messages used in another MAS, and to consider message inputs and outputs as well. This issue will drive our future directions of investigation.

REFERENCES

[1] Euzenat, J., Shvaiko, P.: Ontology matching. Springer (2007)

[2] Ancona, D., Drossopoulou, S., Mascardi, V.: Automatic generation of self-monitoring MASs from multiparty global session types in Jason. In: DALT. Volume 7784 of LNCS., Springer (2012) 76–95

[3] Ancona, D., Barbieri, M., Mascardi, V.: Constrained global types for dynamic checking of protocol conformance in multi-agent systems. In: SAC, ACM (2013) 1377–1379

[4] Briola, D., Mascardi, V., Ancona, D.: Distributed runtime verification of JADE multiagent systems. In: IDC. Volume 570 of Studies in Computational Intelligence., Springer (2014) 81–91

[5] Ancona, D., Briola, D., Ferrando, A., Mascardi, V.: Global protocols as first class entities for self-adaptive agents. In: AAMAS, ACM (2015) 1019–1029

[6] Ancona, D., Bono, V., Bravetti, M., Campos, J., Castagna, G., Deniélou, P., Gay, S.J., Gesbert, N., Giachino, E., Hu, R., Johnsen, E.B., et al., F.M.: Behavioral types in programming languages. Foundations and Trends in Programming Languages **3**(2-3) (2016) 95–230

[7] Ancona, D., Ferrando, A., Mascardi, V.: Comparing trace expressions and linear temporal logic for runtime verification. In: Theory and Practice of Formal Methods. Volume 9660 of LNCS. (2016) 47–64

[8] Ancona, D., Ferrando, A., Mascardi, V.: Parametric runtime verification of multiagent systems. In: AAMAS, ACM (2017) 1457–1459

[9] Ancona, D., Ferrando, A., Franceschini, L., Mascardi, V.: Parametric trace expressions for runtime verification of Java-like programs. In: FTfJP@ECOOP, ACM (2017) 10:1–10:6

[10] Bellifemine, F.L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. Wiley (2007)

[11] The OASIS Web Services Business Process Execution Language (WS-BPEL) Technical Committee: Web services business process execution language version 2.0. (2007)

[12] Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y., Barreto, C.: Web Services Choreography Description Language v. 1.0. (2005)

[13] Baldoni, M., Baroglio, C., Martelli, A., Patti, V., Schifanella, C.: Verifying the conformance of web services to global interaction protocols: A first step. In: EPEW/WS-FM. Volume 3670 of LNCS. (2005) 257–271

[14] Baldoni, M., Baroglio, C., Martelli, A., Patti, V., Schifanella, C.: Verifying protocol conformance for logic-based communicating agents. In: CLIMA. Volume 3487 of LNCS. (2004) 196–212

[15] Baldoni, M., Baroglio, C., Martelli, A., Patti, V.: Verification of protocol conformance and agent interoperability. In: CLIMA. Volume 3900 of LNCS. (2005) 265–283

[16] Baldoni, M., Baroglio, C., Martelli, A., Patti, V.: A priori conformance verification for guaranteeing interoperability in open environments. In: ICSOC. Volume 4294 of LNCS. (2006) 339–351

[17] Baldoni, M., Baroglio, C., Chopra, A.K., Desai, N., Patti, V., Singh, M.P.: Choice, interoperability, and conformance in interaction protocols and service choreographies. In: AAMAS (2), IFAAMAS (2009) 843–850

[18] Baldoni, M., Baroglio, C., Capuzzimati, F.: Typing multi-agent systems via commitments. In: EMAS@AAMAS. Volume 8758 of LNCS. (2014) 388–405

[19] Baldoni, M., Baroglio, C., Capuzzimati, F., Micalizio, R.: Type checking for protocol role enactments via commitments. Autonomous Agents and Multi-Agent Systems **32**(3) (2018) 349–386

[20] Yolum, P., Singh, M.P.: Commitment machines. In: ATAL. Volume 2333 of LNCS. (2001) 235–247

[21] Baldoni, M., Baroglio, C., Capuzzimati, F.: 2COMM: A commitment-based MAS architecture. In: EMAS@AAMAS. Volume 8245 of LNCS. (2013) 38–57

[22] Omicini, A., Ricci, A., Viroli, M.: Artifacts in the A&A meta-model for multi-agent systems. Autonomous Agents and Multi-Agent Systems **17**(3) (2008) 432–456

[23] Ricci, A., Piunti, M., Viroli, M.: Environment programming in multi-agent systems: an artifact-based perspective. Autonomous Agents and Multi-Agent Systems **23**(2) (2011) 158–192

[24] Bordeaux, L., Salaün, G., Berardi, D., Mecella, M.: When are two web services compatible? In: TES. Volume 3324 of LNCS. (2004) 15–28

[25] Busi, N., Gorrieri, R., Guidi, C., Lucchi, R., Zavattaro, G.: Choreography and orchestration: A synergic approach for system design. In: ICSOC. Volume 3826 of LNCS. (2005) 228–240

[26] Bravetti, M., Zavattaro, G.: Contract based multi-party service composition. In: FSEN. Volume 4767 of LNCS. (2007) 207–222

[27] Bravetti, M., Zavattaro, G.: A theory for strong service compliance. In: COORDINATION. Volume 4467 of LNCS. (2007) 96–112

[28] Endriss, U., Maudet, N., Sadri, F., Toni, F.: Protocol conformance for logic-based agents. In: IJCAI, Morgan Kaufmann (2003) 679–684

[29] Alberti, M., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: The $\mathcal{S}$ciff abductive proof-procedure. In: AI*IA. Volume 3673 of LNCS. (2005) 135–147

[30] Alberti, M., Gavanelli, M., Lamma, E., Chesani, F., Mello, P., Torroni, P.: Compliance verification of agent interaction: a logic-based software tool. Applied Artificial Intelligence **20**(2-4) (2006) 133–157

[31] Chopra, A.K., Singh, M.P.: Producing compliant interactions: Conformance, coverage, and interoperability. In: DALT. Volume 4327 of LNCS. (2006) 1–15

[32] Chopra, A.K., Singh, M.P.: Interoperation in protocol enactment. In: DALT. Volume 4897 of LNCS. (2007) 36–49

[33] Giordano, L., Martelli, A.: Verifying agent conformance with protocols specified in a temporal action logic. In: AI*IA. Volume 4733 of LNCS. (2007) 145–156

[34] Ancona, D., Dovier, A.: A theoretical perspective of coinductive logic programming. Fundam. Inform. **140**(3-4) (2015) 221–246

[35] Ferrando, A.: The early bird catches the worm: first verify, then monitor! Presented at Vortex'16 (2016)

[36] Ancona, D., Briola, D., El Fallah Seghrouchni, A., Mascardi, V., Taillibert, P.: Efficient verification of MASs with projections. In: EMAS@AAMAS. Volume 8758 of LNCS. (2014) 246–270

[37] Mascardi, V., Locoro, A., Rosso, P.: Automatic ontology matching via upper ontologies: A systematic evaluation. IEEE Trans. Knowl. Data Eng. **22**(5) (2010) 609–623