A new framework for knowledge revision of abductive agents through their interaction (preliminary report)*

Andrea Bracciali[1] and Paolo Torroni[2]

[1] Dipartimento di Informatica, Università di Pisa
Via Buonarroti, 2 - 56127 Pisa, Italy
braccia@di.unipi.it
[2] DEIS, Università di Bologna
Viale Risorgimento, 2 - 40136 Bologna, Italy
ptorroni@deis.unibo.it

Abstract. The aim of this work is the design of a framework for the revision of knowledge in abductive reasoning agents, based on interaction. We address issues such as: how to exploit knowledge multiplicity to find solutions to problems that agents could not individually solve, what information must be passed or requested, how can agents take advantage from the answers that they obtain, and how can they revise their reasoning process as a consequence of interacting with each other. In this preliminary report, we describe a novel negotiation framework in which agents will exchange not only abductive hypotheses but also meta-knowledge, such as their own integrity constraints. Besides, we formalise some aspects of such a framework, by introducing an algebra of integrity constraints, aimed at formally supporting the updating/revising process of the agent knowledge.

1 Multiple-source knowledge and coordinated reasoning

The agent metaphor has recently become a very popular way to model distributed systems, in many application domains that require a goal directed behaviour of autonomous entities. Thanks also to the recent explosion of the Internet and communication networks, the increased accessibility of knowledge located in different sources at a relatively low cost is opening up interesting scenarios where communication and knowledge sharing can be a constant support to the reasoning activity of agents. In knowledge-intensive applications, the agent paradigm will be able to enhance traditional stand-alone expert systems interacting with end-users, by allowing for inter-agent communication and autonomous revision of knowledge.

Some agent-based solutions can be already found in areas such as information and knowledge integration (see the Sage and Find Future projects by Fujitsu),

* This work is partially funded by the Information Society Technologies programme of the European Commission under the IST-2001-32530 SOCS Project [1].

Business Process Management (Agentis Software), the Oracle Intelligent Agents, not to mention decentralized control and scheduling, and e-procurement (Rockwell Automation, Living Systems AG, Lost Wax, iSOCO), just to cite some.[3]

In order to make such solutions reliable, easy to control, to specify and verify, and in order to make their behaviour easy to understand, sound and formal foundations are needed. For this reason, recent work in logic programming considers multi-agent systems as an interesting and powerful paradigm. Work done by Kowalski and Sadri [2] on the agent cycle, by Leite et al. [3] on combining several Non-Monotonic Reasoning mechanisms in agents, by Satoh et al. [4] on speculative computation, by Dell'Acqua et al. [5] on agent communication and updates, by Sadri et al. [6] on agent dialogues, and by Ciampolini et al. [7] on the coordination of reasoning of abductive logic agents, are only some examples of application of Logic Programming techniques to Multi-Agent Systems. A common characteristic among them is that the agent paradigm brings about the need for dealing with knowledge incompleteness (due to the multiplicity and autonomy of agents), and evolution (due to their interactions).

In this research effort, many proposals have been put forward that consider negotiation and dialogue a suitable way to let agents exchange information and solve problems in a collaborative way, and that consider abduction as a privileged form of reasoning under incomplete information. However, such information exchange is often limited to simple facts that help agents revise their beliefs. In [7], for instance, such facts are modelled as hypotheses made to explain some observation in a coordinated abductive reasoning activity; in [4] the information exchanged takes the form of answers to questions aimed at confirming/disconfirming assumptions; in [6] of communication acts in a negotiation setting aimed at sharing resources. [5] and previous work of the same authors present a combination of abduction and updates in a multi-agent setting, where agents are able to propose updates to the theory of each other in different patterns. In this scenario of collaboration among abductive agents, we envisage a framework in which agents are able to exchange knowledge in various ways. Our idea is to define a framework where agents can exchange information in the form of predicates, theories, integrity constraints, and they do it as a result of a negotiation process. Negotiation is therefore about knowledge. Agents will be able to revise their own constraints, for example by relaxing or tightening them.

In this way, the revision mechanism strongly exploits the interaction between agents. This paper describes preliminary work, where we focus especially on information exchange about integrity constraints. In doing so, we abstract away from issues such as ontology and communication languages and protocols, and we assume that all agents have a common ontology, and that they communicate by using the same language. Agents will actively ask for pieces of knowledge, let them be facts, hypotheses or integrity constraints, and agents will autonomously

[3] A collection of excerpts of papers and web pages about industrial applications of agent technology, including references to the above mentioned projects and applications, can be downloaded from the address: `http://lia.deis.unibo.it/~pt/misc/AIIA03-review.pdf`.

decide whether and how to modify their own constraints whenever it is needed. For instance, an agent, which is unable to explain some observation given its current knowledge, will try to collect information from other agents, in the forms mentioned above, and possibly decide to relax its own constraints in a way that allows him to explain such an observation. Conversely, an agent may find out that some assumptions that he made ought to be inconsistent (for instance, due to social constraints), and try to gather information about how to tighten its own constraints or add new ones which prevent him from making such assumptions.

The distinguishing features of the distributed reasoning revision methodology that we envisage consist of a mix of introspection capabilities and communication capabilities. In the style of abductive reasoning, agents are able to provide conditional explanations about the facts that they prove, they are able to communicate such explanations in order to let others validate them, and they are able to single out and communicate the constraints that prevent from or allow them to explain an observation. Finally, agents are able to revise their constraints, according to those that may be proposed by others.

Below, we informally present a possible interaction among two agents which leads them to modify their knowledge in different ways. This example will be elaborated later, after that the necessary notation is introduced.

Example 1. Let us consider an interaction among two agents, A and B, having different expertise about a given topic.

(1) $A \not\models \neg f, b$ - Agent A is unable to prove (find an explanation for) the goal (observation) "there is a *b*ird that does not *f*ly"...

(2) $A \rightarrow B : \neg f, b$ - ... hence it *asks* agent B for an explanation ...

(3) $B \rightarrow A : \neg f, b \;\; \Delta$ - ... which B returns, as a set Δ of assumptions which explain $\neg f, b$ (e.g., $\Delta = \{p\}$: a *p*enguin is a bird that does not fly);

(4) $A \rightarrow B : IC_\Delta$ - A, driven by the assumptions suggested by B, is able to *determine* a *(significant)* set IC_Δ of constraints that prevented him from assuming $\{\neg f, b, p\}$, and therefore from proving the goal (e.g., $\{\neg f, b \rightarrow false\}$: it is not possible that a bird does not fly). A will *communicate to* B the set IC_Δ;

(5) $B \rightarrow A : \overline{IC_\Delta}$ - B is *able to find* a set $\overline{IC_\Delta}$ that *relaxes* the original constraints IC_Δ and *has been used* in the proof for the goal by B, e.g. $\{\neg f, b, \neg p \rightarrow false\}$ (it is not possible that a bird does not fly, unless it is a penguin). B *proposes* $\overline{IC_\Delta}$ to A;

(6) $A_{\ominus IC_\Delta \oplus \overline{IC_\Delta}} \models \neg f, b$ - A *revises* its constraints with those proposed by B, by means of some *updating operations*, and it is able now to prove its goal.

This brief example hides several non-trivial steps that involve deduction, introspection, interaction, and revision. With this aim, we initially discuss these aspects to illustrate the global picture, and then we focus on the integrity con-

straint revision process (understood within the overall framework). The long-term goal of our research is the definition of the complete framework for distributed revision of agent knowledge.

The rest of this paper is organised as follows: Section 2 introduces Abductive Logic Programming, Section 3 discusses the above mentioned open points. Section 4 presents the formal model we devised to address these open points; there we define an algebra of constraints, constraint selection operators allowing one to determine which constraints are relevant in a proof, and constraint updating/revising operators. Possible applications of the framework are illustrated in Section 5. Concluding remarks and future work are summarised in Section 6.

2 Background on Abductive Logic Programming

In this section we give some background on Abductive Logic Programming and we introduce some notation about abductive agents.

An Abductive Logic Program (ALP) is a triple $\langle T, \mathcal{A}, \mathcal{IC}\rangle$, where T is a theory (a set of predicate definitions), $\mathcal{A}$ is a set of predicates that we call "abducibles", and $\mathcal{IC}$ is a set of integrity constraints. Given an ALP $\langle T, \mathcal{A}, \mathcal{IC}\rangle$ and a goal G, the initial goal or "observation", *abduction* is the process of determining a set Δ of abducible predicates ($\Delta \subseteq \mathcal{A}$), such that (the symbol $\models$ denotes entailment):

$$T \cup \Delta \models G, \text{ and}$$
$$T \cup \mathcal{IC} \cup \Delta \not\models \bot.$$

If such a set exists, we call Δ an abductive "explanation" for G in $\langle T, \mathcal{A}, \mathcal{IC}\rangle$:

$$\langle T, \mathcal{A}, \mathcal{IC}\rangle \overset{abd}{\models}_{\Delta} G$$

Abduction is reasoning in presence of uncertainty, represented as the possible abducibles that can be assumed in order to explain an observation. In this context, the set of integrity constraints $\mathcal{IC}$ of an ALP determines the assumptions which can coherently be made together. Informally speaking, $\mathcal{IC}$ limits the choice of possible explanations to observations, or in other words, it rules out some hypothetical worlds from those modelled by a given ALP.

2.1 Syntax and Semantics of Integrity Constraints

We consider integrity constraints having the following syntax:

$$\begin{aligned} ic &::= \bot \leftarrow Body \\ Body &::= Literal\ [,\ Literal] \\ Literal &::= Atom \mid \neg Atom \end{aligned}$$

Constraints of the form: $\bot \leftarrow x, \neg x$ are left *implicit* in the agents' abductive logic programs, and they can be used to implement default negation.[4]

[4] According to [8], negation as default can be recovered into abduction by replacing negated literals of the form $\neg\ a$ with a new positive, abducible atom not_a and by adding the integrity constraint $\leftarrow a, not\_a$ to $\mathcal{IC}$.

Given an integrity constraint $ic = \bot \leftarrow L_1, \ldots, L_n$, we call $body(ic)$ the set of literals $\{L_1, \ldots, L_n\}$. Also, we denote singleton integrity constraints by ic, ic_1, ic_2, ic', ic'', ..., and sets of integrity constraints by $\mathcal{IC}$, $\mathcal{IC}_1$, $\mathcal{IC}_2$, ..., $\mathcal{IC}'$, $\mathcal{IC}''$, ... Finally, we denote sets of literals by Δ, Δ_1, Δ_2, ..., Δ', Δ'', ... In the following, we will adopt the notation $\leftarrow L_1, \ldots, L_n$ as a shorthand for $\bot \leftarrow L_1, \ldots, L_n$.

Intuitively, an integrity constraint $ic = \leftarrow L_1, \ldots, L_n$ represents a restriction, preventing $L_1, \ldots, L_n$ from being all true all at the same time. If some of the L_j in the body of ic are abducible atoms, ic constrains the set of possible explanations that can be produced during an abductive derivation process.

Example 2. Let us consider agent A of Example 1. Its abductive program states that something can be considered a bird if either it flies or it has *fe*athers, while it can be considered a *m*ammal if it has *ha*ir. Finally, a *d*olphin is something that *s*wims and has no hair:

$$T = \left\{ \begin{array}{l} b \leftarrow fe. \\ b \leftarrow f. \\ m \leftarrow ha. \\ d \leftarrow s, \neg ha. \end{array} \right\} \qquad \mathcal{A} = \{f, s, fe, ha\} \qquad \mathcal{IC} = \left\{ \begin{array}{l} \leftarrow b, \neg f \\ \leftarrow d, \neg s \end{array} \right\}$$

In order to explain its observations, agent A can make assumptions according to its set $\mathcal{A}$, and, for instance, categorise a dolphin by assuming that it is able to swim. Note how abducibles have no definitions. $\mathcal{IC}$ prevents A from having a bird that does not fly or a dolphin that does not swim (together with all the implicit integrity constraints). It is clear that there is no $\Delta \subseteq \mathcal{A}$ such that $T \cup \Delta \models b, \neg f$ and $T \cup \Delta \cup \mathcal{IC} \not\models \bot$, and hence, as supposed in point (1) of Example 1 (in its informal language), $A \not\models b, \neg f$.

We consider *abductive agents* to be agents whose knowledge is represented by an abductive logic program, provided with an abductive entailment operator. At this stage, we do not make assumptions on the underlying operational model.

3 Issues in a constraint revision framework

Example 1 has informally introduced some issues which a framework supporting a distributed constraint revision process must address. In this section, we discuss them in more detail. The next section will provide a better technical insight, by presenting some preliminary results about how some of these issues could be formally modeled.

3.1 Communication

Points (2), (3), (4) and (5) of Example 1 require communication between agents. We are aware that the literature on this topic is huge. There is a large number

of existing proposals for agent communication models and languages, several dedicated discussion forums such as the Workshop on Agent Communication Languages and Protocols [9,10], and several conference tracks and special journal issues on this topic. Here, we do not intend to propose a new communication language or a new interaction model, but we assume that agents are able to communicate according to some language and protocols, and that they have a common ontology. The focus of the present work is rather on the *content* than on the syntax, semantics, or pragmatics of the messages that agents exchange with each other, or on the protocols which support at a lower level the knowledge revision process that we envisage. In this section, we make some remarks about communication in our framework from this perspective.

Assumption and constraint exchange. Point (3) requires that agents are able to exchange the sets of assumptions they make. In general, existing abductive agent frameworks deal with assumptions at a semantic level, but do not consider assumptions as a first order object in the content of a message. For instance, in [6], the messages themselves are modelled as abducible predicates, but they do not predicate about assumptions. In the computational model of [7], when agents cooperatively resolve a query, the computational model is in charge of checking the global consistency of the assumptions made against the integrity constraints of all the agent, but, still, agents can not, for instance, directly ask other agents to check for the consistency of a given set of abducibles.

This enhancement with respect to the previous work done on the subject is necessary in order to exploit assumptions in the process of determining significant sets of constraints, like in points (3) and (4). In particular, agents can be required to explain an observation, starting from a given set of assumptions, as A is doing in points (3) and (4), determining the set IC_{Δ}. Similarly, it is also necessary to communicate sets of constraints, points (4) and (5), which is not allowed in existing abductive agent frameworks either, to the best of our knowledge, and for which analogous considerations hold.

Identity discover. Starting from point (2), agent A establishes a conversation with agent B. This requires a mechanism allowing A to select B to speak with. For instance, this could be accomplished by means of access mechanisms to semi-open societies [11], such as facilitators or directory servers which let A meet its possible partners. This kind of problem is not addressed in this paper. Instead, we will restrict ourselves to a two-party setting, where agents are already aware of each other.

Trust and agent hierarchies. Almost all interaction described in Example 1 is based on relations of trust or dependency among agents, which could be based on some form of reputation [12] or it can be determined by social ties such as those given by a hierarchy. In particular, in the example, agent A and B expose their assumptions and even their constraints, and agent A revises its knowledge based on the view of the world of B. Similarly to the previous point, we do not discuss here the mechanisms which justify such relations. The hypothesis that cooperating agent will share parts of their knowledge is an assumption that we have in common with other work of literature, such as [5] and [7].

Social constraints. We would like to make a final remark about the use of communicating constraints within a society. Although integrity constraints and assumptions are specific of abductive agents, in [13] an interaction framework is proposed where integrity constraints are used to model interaction protocols. Such protocols are not inside the agents but in a social infrastructure which is outside the agents. In this setting, a society where certain interaction protocols are defined by way of integrity constraints could notify such constraints to newcomers, in order to help them behave according to the protocols. Or else, a society could notify protocol updates to its members.

3.2 Proof of goals and relevance of integrity constraints

Constraint revision, as illustrated in Example 1, requires the capability to determine a set of constraints which are relevant for a given computation, either in the sense that they contribute to defining a set of assumptions in a successful proof, or because they make a proof fail. For instance, in point (4), A, exploiting the explanation Δ provided by B, determines the set IC_Δ of integrity constraints which prevent A from explaining its observation by the set of assumptions Δ itself. Let $\Delta = \{p, \neg f\}$. While the problem of determining significant integrity constraints for a given proof is in general hard, we believe that it could be computationally viable to identify the set $IC_\Delta = \{\leftarrow b, \neg f\}$ as relevant for the failure of the query $b, \neg f$,given Δ (for instance, by exploring the proof of $b, \neg f, p$, i.e. the proof of the initial goal together with Δ). The definition of a proof procedure supporting similar analysis is one of our ongoing activities.

¿From a declarative perspective, relevant sets for a successful proof are formally defined in Section 4.1. This definition is used, for instance, by agent B in determining the set $I\bar{C}_\Delta = \{\leftarrow b, \neg f, \neg p\}$ which is relevant for the proof of the query, and relaxes (see below) the set IC_Δ, points (4) and (5) of the example.

In general, there will be several alternative sets of relevant constraints for the same query, e.g., because there are different alternative proofs for it. The existence of a unique minimal set is not guaranteed either. For this reason, we envisage that the knowledge revision process will have to go through a trial-and-error search process for a suitable minimal sets. For instance, agents can iterate a cooperative dialog, similar to that of Example 1, in which they propose to each other a number of minimal significant sets, until they converge to a successful proof, if there exists any. This is in line with the approach of [7] and [6]. In Section 5, we illustrate a possible cycle which allows for such an interaction in the form of a dialog.

3.3 Constraint hierarchies

Point (5) requires both the definition of the concept of *relaxation* of a set of constraints, and the computational capability to determine, given a set $\mathcal{IC}$, another set $\mathcal{IC}'$ that relaxes it. In order to address the former issue, we propose an algebra of $\mathcal{IC}$s, presented in Section 4.2, provided with a relaxation-based partial order, while the latter issue has not yet been fully addressed.

Intuitively, a set $\mathcal{IC}'$ of integrity constraints *relaxes* a set $\mathcal{IC}$, $\mathcal{IC} <^{IC} \mathcal{IC}'$, if all the goals that can be proved with respect to $\mathcal{IC}$ can be also proved with respect to $\mathcal{IC}'$ (but not vice-versa).

The relaxing relation, in some basic cases, can be checked by means of a simple necessary condition, namely a syntactical comparison. This happens when the relations is the set inclusion partial order. More in general, the relation may fail to be a partial order, in which case it is a partial pre-order, and checking whether a set relaxes another one according to the definition of relaxing relation may require more complex analysis, based on the semantics of the corresponding ALP. This point is currently under investigation.

Conversely, the partial order can be also read as a tightening-based partial order, i.e. $\mathcal{IC}'$ *tightens* $\mathcal{IC}$, $\mathcal{IC}' <^{IC} \mathcal{IC}$, when the goals that can be proved with respect to $\mathcal{IC}'$ are a subset of those that can proved with respect to $\mathcal{IC}$. Hence, the same formal model can be used to describe interactions, which, possibly depending on the issues discussed in Section 3.1, may lead an agent to restrict his own believes.

Finally, the partially ordered set of integrity constraints can be equipped with operations for relaxing, tightening and more in general revising sets of integrity constraints, as described in the next Section 3.4.

3.4 Constraint revision

Once that a relaxing $\mathcal{IC}'$ has been found, it is necessary to be able to use it in order to revise the constraint set $\mathcal{IC}$ of an agent, as it happens in point (6), where A revises its own integrity constraints.

In general, only a subset of the integrity constraints of an agent will have to be revised according to a relaxing or tightening operation. To this aim, we have defined a relaxation operator which, given two sets of integrity constraints $\mathcal{IC}_a$ and $\mathcal{IC}_b$ returns a third set of integrity constraints, $\mathcal{IC}' = \mathcal{IC}_a \biguplus \mathcal{IC}_b$ which relaxes $\mathcal{IC}_a$ in case $\mathcal{IC}_a <^{IC} \mathcal{IC}_b$. The definition of $\biguplus$ can be found in Section 4.

The operator $\biguplus$ is only a possibility that we considered. The definition of other tightening and revising operations is under current investigation.

Example 3. After having explained in more detail the steps for constraint revision, it is now possible to revise Example 1, according to the program of A introduced in the Example 2 and the above discussed issues.

(1) $A \overset{abd}{\not\models} \neg f, b$ -
(2) $A \rightarrow B : \neg f, b$ -
(3) $B \rightarrow A : \neg f, b \;\; \{\neg f, p\}$ - $\{\neg f, p\}$ is the abductive explanation provided by B.
(4) $A \rightarrow B : \{\leftarrow b, \neg f\}$ - $\{\leftarrow b, \neg f\}$ is the significant set of constraints that prevents A from proving the query based on B's assumptions

(5) $B \rightarrow A : \{\leftarrow b, \neg f, \neg p\}$ - B has generated $\{\leftarrow b, \neg f, \neg p\}$ as relaxation for the significant set provided by A.

(6) $\mathcal{IC}'_A = \left\{ \begin{array}{l} \leftarrow b, \neg f, \neg p \\ \leftarrow d, \neg s \end{array} \right\}$ $A \overset{abd}{\models}_{\Delta} \neg f, b$ - After the update, the relaxed $\mathcal{IC}'_A$ is shown. A is able to prove its goal (We can safely assume that $p \in \mathcal{A}_A$).

4 A formal model for constraint revision

In this section we illustrate some of the technical aspect we devised in order to support the model for constraint revision that we informally discussed in the previous section. Firstly, we define the set of integrity constraints that are significant in a successful or failed proof; secondly, we present a partial order of constraint sets according to a relaxing relation (that can be dually read as a tightening relation); finally, we introduce an updating operator for revising a (significant) set of constraints, which is based on the partial order presented.

4.1 Determining significant sets of constraints

Definition 1 characterises a minimal set of the constraints which have been involved in a successful abductive proof, producing the minimal set of explanations Δ. All the constraints in the set are directly involved in the proof, indeed, as soon as one of them is removed, a smaller set of explanations than the one supposed to be minimal can be produced by a successful proof for the same goal (i). Moreover, the set is required to be minimal (ii).

Definition 1. *Let $P = \langle T, \mathcal{A}, \mathcal{IC} \rangle$ be an abductive logic program and G a goal, such that $\langle T, \mathcal{A}, \mathcal{IC} \rangle \overset{abd}{\models}_{\Delta} G$ (i.e., Δ is a possible explanation for G in P). Let us assume that Δ is minimal, i.e. $\nexists\, \hat{\Delta} \subset \Delta$ such that $\langle T, \mathcal{A}, \mathcal{IC} \rangle \overset{abd}{\models}_{\hat{\Delta}} G$.*[5] *A set of integrity constraints $\mathcal{IC}' \subseteq \mathcal{IC}$ is* relevant *with respect to P and G, written $\mathcal{IC}'$ **rel** $(\langle T, \mathcal{A}, \mathcal{IC} \rangle, G)$, $\overset{def}{\Longleftrightarrow}$*

$$\begin{array}{ll} & \exists \Delta' \subset \Delta \text{ such that } \langle T, \mathcal{A}, \mathcal{IC} \setminus \mathcal{IC}' \rangle \overset{abd}{\models}_{\Delta'} G, \text{ and} \quad \text{(i)} \\ (\mathcal{IC}' \text{ is minimal}) & \nexists\, \mathcal{IC}'' \subset \mathcal{IC}' \text{ such that } \mathcal{IC}'' \ \textbf{\textit{rel}}\ (\langle T, \mathcal{A}, \mathcal{IC} \rangle, G). \quad \text{(ii)} \end{array}$$

Example 4. Let us consider agent B of Example 1. It has a simple abductive program, stating that something can be considered a bird if either it flies or it is a penguin.

$$T = \left\{ \begin{array}{l} b \leftarrow f. \\ b \leftarrow p. \end{array} \right\} \qquad \mathcal{A} = \{f, p\} \qquad \mathcal{IC} = \{\leftarrow b, \neg f, \neg p\}$$

[5] Otherwise the definition can be recast on the (supposed minimal) Δ'.

Agent B can assume that something flies or it is a penguin, according to its set $\mathcal{A}$, while $\mathcal{IC}$ prevents A from having a bird that does not fly unless it is a penguin. Trivially, $\langle T, \mathcal{A}, \mathcal{IC}\rangle \overset{abd}{\models}_{\{\neg f,p\}} \neg f, b$, and $\{\neg f, p\}$ is minimal. Moreover, $\mathcal{IC} = \{\leftarrow b, \neg f, \neg p\}$ **rel** $(\langle T, \mathcal{A}, \mathcal{IC}\rangle, \ \neg f, b)$, since

(i) $\langle T, \mathcal{A}, \mathcal{IC} \setminus \mathcal{IC} = \emptyset\rangle \overset{abd}{\models}_{\emptyset} \neg f, b$, and
(ii) $\nexists\, \mathcal{IC}'' \subset \mathcal{IC}$ such that $\mathcal{IC}''$ **rel** $(\langle T, \mathcal{A}, \mathcal{IC}\rangle, \ \neg f, b)$.

The previous example shows how Definition 1 characterises the set of constraints that B is able to propose to A in point (5) of our Example 1.

On the other hand, given a failing derivation $\langle T, \mathcal{A}, \mathcal{IC}\rangle \overset{abd}{\not\models} G$ of an abductive agent, it is also worth giving a characterisation of a (minimal) subset of $\mathcal{IC}$ such that, once relaxed, or removed, allows G to be proved. This is the notion of *minimally relaxing* set defined by Definition 2. For an example of minimally relaxing set see Example 8

Definition 2. *Let $P = \langle T, \mathcal{A}, \mathcal{IC}\rangle$ be an abductive logic program, and G a goal such that $\langle T, \mathcal{A}, \mathcal{IC}\rangle \overset{abd}{\not\models} G$ (i.e., there is no explanation for G in P). A set of integrity constraints $\mathcal{IC}' \subseteq \mathcal{IC}$ is* minimally relaxing *P (towards explaining G), written $\mathcal{IC}'$ **min_relax** $(\langle T, \mathcal{A}, \mathcal{IC}\rangle, G)$, $\overset{def}{\iff}$*

$$\exists \Delta \text{ such that } \langle T, \mathcal{A}, \mathcal{IC} \setminus \mathcal{IC}'\rangle \overset{abd}{\models}_{\Delta} G, \text{ and} \quad \text{(iii)}$$

$$(\mathcal{IC}' \text{ is minimal}) \ \nexists\, \mathcal{IC}'' \subset \mathcal{IC}' \text{ s.t. } \mathcal{IC}'' \ \textbf{\textit{min\_relax}}\ (\langle T, \mathcal{A}, \mathcal{IC}\rangle, G) \quad \text{(iv)}$$

4.2 A relaxing-tightening hierarchy of constraints

In this section, we define a partial ordering relationship among integrity constraints, and then we lift it to sets of integrity constraints. For this purpose, we introduce a symbol $<^{IC}$(to read: "more restrictive than", or "less relaxed than"). This (complete, under reasonable hypothesis) partial ordering is a specific instantiation of the general notions of relaxation and tightening applied to set of constraints, and it will be used in the next section for defining the operators of revision.

Definition 3. *Given a pair of integrity constraints, ic_i and ic_j,*

$$ic_i <^{IC} ic_j \quad \overset{def}{\iff} \quad \forall \Delta, \Delta \cup \{ic_i\} \not\models \bot \Rightarrow \Delta \cup \{ic_j\} \not\models \bot.$$

Example 5. Let us consider the following constraints:

(ic_1) $\bot \leftarrow a, b,$
(ic_2) $\bot \leftarrow a, b, c,$

Then, $ic_1 <^{IC} ic_2$.

We can generalize the ordering relationship, from pairs of integrity constraints to pairs of sets of integrity constraints.

Definition 4. *Given two sets of integrity constrains, $\mathcal{IC}_i$ and $\mathcal{IC}_j$,*

$$\mathcal{IC}_i <^{IC} \mathcal{IC}_j \quad \stackrel{def}{\iff} \quad \forall \Delta, \Delta \cup \mathcal{IC}_i \not\models \bot \Rightarrow \Delta \cup \mathcal{IC}_j \not\models \bot.$$

The ordering initially introduced between pairs of integrity constraints is a special case of the more general ordering relation between pairs of sets. In particular, from Definitions 3 and 4 it follows that:

Lemma 1. *Given a pair of integrity constraints, ic_i and ic_j,*

$$body(ic_i) \subset body(ic_j) \Leftrightarrow ic_i <^{IC} ic_j \tag{v}$$
$$\{ic_i\} <^{IC} \{ic_j\} \Leftrightarrow ic_i <^{IC} ic_j \tag{vi}$$

If $\mathcal{IC}_j$ is constituted by a single element, $\mathcal{IC}_j = \{ic_j\}$, we use the notation $\mathcal{IC}_i <^{IC} ic_j$ to denote $\mathcal{IC}_i <^{IC} \{ic_j\}$.

Example 6. By defining the *zero element* of the $<^{IC}$ relation as $ic^0 \quad \stackrel{def}{\iff} \quad \bot \leftarrow$, i.e. the integrity constraint which is never satisfied, it is easy to see that

$$\forall ic.\ ic^0 <^{IC} ic.$$

Similarly, the *one element* is the constraint whose body is satisfied only if all the (finitely many) abducibles have been assumed: $ic^1 \quad \stackrel{def}{\iff} \quad \bot \leftarrow \bigwedge_i a_i\ \forall a_i \in \mathcal{A}$. It holds

$$\forall ic.\ ic <^{IC} ic^1.$$

Let us also consider the following constraints:

(ic_1) $\bot \leftarrow a, b$,
(ic_3) $\bot \leftarrow c, \neg b$,
(ic_4) $\bot \leftarrow a, c$, and the implicit constraint
(ic_5) $\bot \leftarrow b, \neg b$.

Then, $\{ic_1, ic_3, ic_5\} <^{IC} ic_4$.

4.3 A constraint updating operator

We define an "update with relaxation" operator $\mathcal{IC}_a \biguplus \mathcal{IC}_b$, which updates the set of integrity constraints $\mathcal{IC}_a$ with respect to $\mathcal{IC}_b$, relaxing, whenever possible, the constraints in $\mathcal{IC}_a$ that are "less relaxed than" $(<^{IC})$ those in $\mathcal{IC}_b$. More precisely, given two constraints $ic_a \in \mathcal{IC}_a$ and $ic_b \in \mathcal{IC}_b$, if $ic_a <^{IC} ic_b$ then $ic_a \notin \mathcal{IC}_a \biguplus \mathcal{IC}_b$. The next Example 7, illustrates the use of the operator, whose definition is straightforward for the basic cases 1. and 2. Further examples of its usage in Section 5 will make more clear the rationale of its definition, in particular when the set $\mathcal{IC}_b$ has been determined by a collaborative agent as a relaxation for explaining some facts.

Definition 5. *Let $\mathcal{IC}_a$ and $\mathcal{IC}_b$ be two sets of integrity constraints. Then, the update with relaxation of $\mathcal{IC}_a$ by $\mathcal{IC}_b$, denoted as $\mathcal{IC}_a \biguplus \mathcal{IC}_b$, is defined as follows:*

$$\mathcal{IC}_a \biguplus \mathcal{IC}_b \overset{def}{\iff} \mathcal{IC}_a \setminus \mathcal{IC}'_a \cup \mathcal{IC}_b,$$

where $\mathcal{IC}'_a = \bigcup_{\mathcal{IC}''_a \subseteq \mathcal{IC}_a} \mathcal{IC}''_a$ such that $\mathcal{IC}''_a <^{IC}_{(min)} ic_b$ for some $ic_b \in \mathcal{IC}_b$. $\mathcal{IC} <^{IC}_{(min)} ic$ is defined as follows:

$$\mathcal{IC} <^{IC}_{(min)} ic \overset{def}{\iff} \begin{cases} \mathcal{IC} <^{IC} ic \\ \nexists \mathcal{IC}' \subset \mathcal{IC} \text{ such that } \mathcal{IC}' <^{IC} ic \end{cases}$$

We have two special cases of $\mathcal{IC}_a \biguplus \mathcal{IC}_b$:

1. given two integrity constraints, ic_a and ic_b,

$$\{ic_a\} \biguplus \{ic_b\} \iff \begin{cases} \{ic_b\}, & \text{if } ic_a <^{IC} ic_b \\ \{ic_a, ic_b\}, & \text{otherwise} \end{cases}$$

2. given an integrity constraint ic_a and a set of integrity constraints $\mathcal{IC}_b$,

$$\{ic_a\} \biguplus \mathcal{IC}_b \iff \begin{cases} \mathcal{IC}_b, & \text{if } \exists ic_b \in \mathcal{IC}_b \text{ such that } ic_a <^{IC} ic_b \\ \{ic_a\} \cup \mathcal{IC}_b, & \text{otherwise} \end{cases}$$

Example 7.

Given the following constraints:

(ic_1) $\leftarrow a, b.$
(ic_3) $\leftarrow c, \neg b.$
(ic_4) $\leftarrow a, c.$
(ic_5) $\leftarrow b, \neg b.$
(ic_6) $\leftarrow a, b, \neg c.$

The following propositions hold:

(p_1) $ic_1 <^{IC} ic_6$;
(p_2) $\{ic_1, ic_3, ic_5\} <^{IC}_{(min)} ic_4$;
(p_3) $\{ic_6\} = \{ic_1\} \biguplus \{ic_6\}$;
(p_4) $\{ic_1, ic_6\} = \{ic_6\} \biguplus \{ic_1\}$;
(p_5) $\{ic_4\} = \{ic_1, ic_3, ic_5\} \biguplus \{ic_4\}$;
(p_6) $\{ic_3, ic_4, ic_6\} = \{ic_1, ic_3\} \biguplus \{ic_4, ic_6\}$;
(p_7) $\{ic_4, ic_5\} = \{ic_1, ic_3, ic_5\} \biguplus \{ic_4, ic_5\}$;

5 An example of application of our framework

In this section we show a possible instantiation of the general framework that we envisage. We consider, for the sake of simplicity, a 2-agent setting. Agent a is equipped with the abductive logic program $\langle T_a, \mathcal{A}, \mathcal{IC}_a \rangle$; agent b with $\langle T_b, \mathcal{A}, \mathcal{IC}_b \rangle$. Whenever a fails to explain an observation G within its current abductive logic program, it starts an interaction with b, in order to find a suitable relaxation of its integrity constraints that allows him (a) to explain G. On the other hand, b will have to reply to a's request according to some internal policy.

We distinguish between two phases of the interaction between a and b: (i) the process of asking for a piece of information, be it a set of assumptions, a set of integrity constraints, a set of definitions, or a combination of them, and (ii) the process of answering to incoming requests of that kind. As far as the first

point, we imagine that agents will act according to certain internal cycles and policies. For the sake of clarity, and in order to keep the two things separate, we will call *cycle* the sequence of steps related to the activity described as (i), *policy* that related to the activity described as (ii).

5.1 Cycle

We define a predicate $\mathbf{request}(a, b, \mathcal{IC}'_a, G)$ which an agent a will use to request a set of constraints $\mathcal{IC}'_b$ from an agent b.

As we said earlier in Section 3, in general, there will be several alternative sets of relevant constraints for the same query. For this reason, the knowledge revision process will have to go through a trial-and-error search process for suitable minimal sets. For instance, agents can iterate a cooperative dialog, similar to that of Example 1, in which they propose to each other a number of minimal significant sets, until they converge to a successful proof, if any.

Here, we illustrate a possible cycle which allows for such an interaction in the form of a dialog.

Cycle 1 Agent update cycle for an agent a

1: Find a minimal relaxation, $\mathcal{IC}'_a$ **min_relax** $(\langle T_a, \mathcal{A}, \mathcal{IC}_a\rangle, G)$
2: **repeat**
3: Send $\mathbf{request}(a, b, \mathcal{IC}'_a, G)$
4: Wait for $\mathbf{reply}(b, a, \mathcal{IC}'_a, G, \mathcal{IC}'_b)$
5: **repeat**
6: $\mathcal{IC}''_a = \mathcal{IC}_a \uplus \mathcal{IC}'_b$
7: **if** $\langle T_a, \mathcal{A}, \mathcal{IC}''_a\rangle \overset{abd}{\models}_{\Delta} G$ **then**
8: Update $\mathcal{IC}_a$ by $\mathcal{IC}''_a$
9: **else**
10: Send $\mathbf{request}(a, b, \mathcal{IC}'_a, G)$
11: Wait for $\mathbf{reply}(b, a, \mathcal{IC}'_a, G, \mathcal{IC}^1_b)$
12: $\mathcal{IC}'_b \leftarrow \mathcal{IC}^1_b$
13: **end if**
14: **until** $\langle T_a, \mathcal{A}, \mathcal{IC}_a\rangle \overset{abd}{\models}_{\Delta} G$ or $\mathcal{IC}'_b = \emptyset$
15: **if** $\langle T_a, \mathcal{A}, \mathcal{IC}_a\rangle \overset{abd}{\not\models} G$ **then**
16: Find a *new* minimal relaxation, $\mathcal{IC}^1_a$ **min_relax** $(\langle T_a, \mathcal{A}, \mathcal{IC}_a\rangle, G)$
17: $\mathcal{IC}'_a \leftarrow \mathcal{IC}^1_a$
18: **end if**
19: **until** $\langle T_a, \mathcal{A}, \mathcal{IC}_a\rangle \overset{abd}{\models}_{\Delta} G$ or $\mathcal{IC}^1_a = \emptyset$

We will refer to such a cycle as to **Cycle 1**. Its intuitive reading is the following: whenever a has an observation G that he cannot explain, he will:

1. try and find a minimal relaxation $\mathcal{IC}'_a$ for G;

2. ask b whether he can tell him something about it (more details about b's reply are given below);
3. once a receives a **reply** from b, **reply**$(b, a, \mathcal{IC}'_a, G, \mathcal{IC}'_b)$, if a is able to relax his own constraints by means of $\mathcal{IC}'_b$) and therefore explain the observation, he will revise his knowledge accordingly;
4. otherwise, a will keep asking for different sets of integrity constraints until either he manages to explain his observation, or b has no other sets of constraints to communicate to him.

This is in line with the approach of [7] and [6]. In [7], the object of communication are the hypotheses, in the form of predicates. Groups of agents try to explain an observation consistently with the integrity constraints of each other, and re-iterate the whole computational process by proposing different sets of constraints, until a fixpoint is reached. In [6], agents have a similar cycle when they try to obtain the resources that they are missing. The purpose of defining a cycle in this way is to be able to prove some properties about the outcome of the interaction.

Indeed, one of the main motivations of this framework is being able to prove properties about the outcome of agent interaction. In this case, a property that we would like to obtain is: given two agents a and b, the former trying to relax his constraints by adopting a specific cycle (such as **Cycle 1**), the latter responding to a's requests by following a specific policy (such as **Policy 1**), either b is unable to explain a certain observation G, or it is possible for a to revise his own integrity constraints based on b's reply, and finally explain G using his revised knowledge. This is subject for current work.

5.2 Policy

Upon receipt of a's **request**, b will adopt some policy to put together a set of constraints in reply. We use the predicate: **reply**$(b, a, \mathcal{IC}'_a, G, \mathcal{IC}'_b)$ to indicate b's reply to a's **request**. **Policy 1** is one particular policy, defined for an agent b. Its intuitive reading is the following: whenever b receives a message of kind **request**$(a, b, \mathcal{IC}'_a, G)$ from an agent a, b will do the following:

1. try and find an abductive explanation for G;
2. single out a minimal subset $\mathcal{IC}'_b$ of integrity constraints in $\mathcal{IC}_b$, which are relevant for such an explanation, and which b has not yet communicated to a by means of **reply**$(b, a, \mathcal{IC}'_a, G, \mathcal{IC}'_b)$;
3. communicate a such a set $\mathcal{IC}'_b$: **reply**$(b, a, \mathcal{IC}'_a, G, \mathcal{IC}'_b)$

This policy is only one example. Other ones are indeed possible. In particular, while in this work we are concerned with the exchange of information about constraints, it could also be the case that b's explanation is not by any constraint of a's, and that a is unable to explain G only because a is missing some definition which b instead knows.

We are currently studying policies such as **Policy 1**, expressing the semantics of a **reply**, in order to be able to prove results about the evolution of knowledge of agents in some specific settings, as we discussed above.

Policy 1 Agent policy of an agent b for replying to a **request**$(a, b, \mathcal{IC}'_a, G)$

1: **if** $\langle T_b, \mathcal{A}, \mathcal{IC}_b \rangle \overset{abd}{\not\models} G$ **then**
2: $\quad \mathcal{IC}'_b = \emptyset$
3: **else**
4: $\quad \mathcal{IC}'_b$ is a subset of $\mathcal{IC}_b$ with the following properties:
$\quad\quad (i)$ $\mathcal{IC}'_b$ **rel** $(\langle T_b, \mathcal{A}, \mathcal{IC}_b \rangle, G)$;
$\quad\quad (ii)$ $\exists ic_a \in \mathcal{IC}'_a, ic_b \in \mathcal{IC}'_b$ such that $ic_a <^{IC} ic_b$
(potentially, $\mathcal{IC}'_b$ could be exploited to relax some constraints);
$\quad\quad (iii)$ $\mathcal{IC}'_b$ has not been yet communicated to a as a reply to **request**$(a, b, \mathcal{IC}'_a, G)$
(to prevent b from proposing he same $\mathcal{IC}'_b$ twice in reply to the same request of a's);
5: $\quad$ if such a subset cannot be found, then $\mathcal{IC}'_b = \emptyset$
6: $\quad$ **reply**$(b, a, \mathcal{IC}'_a, G, \mathcal{IC}'_b)$
7: **end if**

Example 8. Let us consider again a slight modification of Example 1, where the set of abducible predicates is $\{b, f, p, s\}$ plus all the negated literals, and the abductive logic programs of A and B are as follows (both theories define the predicate $a$nimal):

$$P_A = \langle T_A, \mathcal{A}, \mathcal{IC}_A \rangle \qquad P_B = \langle T_B, \mathcal{A}, \mathcal{IC}_B \rangle$$

$$T_A \quad \{\, a \leftarrow b \,\} \qquad T_B \quad \{\, a \leftarrow b \,\}$$

$$\mathcal{IC}_A \quad \{\, (ic_1) \leftarrow b, \neg f. \,\} \qquad \mathcal{IC}_B \quad \left\{ \begin{array}{ll} (ic_2) & \leftarrow b, \neg f, \neg p. \\ (ic_3) & \leftarrow d, \neg s. \end{array} \right\}$$

Let us assume that both A and B follow **Cycle 1** to request relaxations of constraints and **Policy 1** to answer to incoming requests. Finally, we assume that at some point A makes the observation $G = \{b, \neg f\}$. Then, the behaviour of the system $\{A, B\}$ is the following:

– (1) $P_A \overset{abd}{\not\models} \{\neg f, b\}$. At this point, following **Cycle 1**, A looks for a minimal relaxation of its constraints, and it finds it in $\{ic_1\} = \{\leftarrow b, \neg f\}$:

$$\{\leftarrow b, \neg f\} \textbf{ min\_relax } (P_A, G)$$

– (2) A asks B a set of relevant integrity constraints of his, that are relevant for the explanation of G:

$$\textbf{request}(a, b, \{\leftarrow b, \neg f\}, \{\neg f, b\});$$

– (3) Upon receipt of A's request, according to **Policy 1** B verifies that he is able to prove G: $P_B \overset{abd}{\models}_{\{b, \neg f, p\}} \{b, \neg f\}$. B finds a set of integrity constraints $\mathcal{IC}'_B = ic_2$ that are relevant to A's request:

$$\{\leftarrow b, \neg f, \neg p\} \textbf{ rel } (P_B, \{b, \neg f\}),$$

and indeed $\{\leftarrow b, \neg f\} <^{IC} \{\leftarrow b, \neg f, \neg p\}$. At this point, B replies to A's request by communicating $\mathcal{IC}'_B$:

$$\mathbf{reply}(a, b, \{\leftarrow b, \neg f\}, \{\neg f, b\}, \{\leftarrow b, \neg f, \neg p\});$$

– (4) A generates the set $\mathcal{IC}''_A = \{\leftarrow b, \neg f, \neg p\}$ as a revision of its own $\mathcal{IC}_A$ by $\mathcal{IC}'_B$: $\mathcal{IC}''_A = \mathcal{IC}_A \biguplus \mathcal{IC}'_B$. Since $\langle T_A, \mathcal{A}, \mathcal{IC}''_A \rangle \overset{abd}{\models}_{\{b,\neg f,p\}} \{b, \neg f\}$, A updates his own abductive logic program.

6 Concluding remarks

In this work, we have described a framework for agent interaction where the object of interaction is the agent knowledge at different levels. We have considered the case of abductive agents that are capable to formulate assumptions to explain some observations. The agent knowledge is coded into an abductive logic program, consisting of a theory, a set of abducible predicates, and a set of integrity constraints. Indeed, abduction, which we considered as representing the entire agent knowledge, could be part of a more comprehensive agent architecture, like that of computees [1], which allows for multiple forms of reasoning and knowledge representation within an agent.

In this work, we discuss about the characteristics of an interaction framework that can be used to exploit the abductive reasoning of a possibly more complex agent. Agents exchange information about the assumptions that they make in order to explain observations, and the integrity constraints used during their reasoning. Integrity constraints "shape" the reasoning processes of agents by ruling out the set of explanation which are considered inconsistent. Exchanging constraints corresponds to revise the reasoning process. The methodology presents some difficulties that we singled out, like understanding the role of sets of constraints in a proof, defining constraint revising mechanisms, or design suitable interaction schemata for the collaboration of agents within a society.

We identified some issues that must be tackled by such a framework, both with respect to the kind of communication required and the kind of agent reasoning involved in the revision process. We instantiated the general framework by choosing a specific syntax for abductive logic programs, defining a partial ordering relation among sets of integrity constraints, and proposing a specific revision operator. Our choices have been motivated by a running example inspired from literature work on nonmonotonic reasoning. In the last section of this paper, we show some ongoing work that we are doing towards applying the framework to specific interaction patterns, and we comment on the kind of properties that we want the framework to exhibit. An issue that will have to be investigated is the computational complexity of such interaction patterns, both for what concerns (i) the computational cost of proceeding in the protocol (i.e., by finding a minimal relaxation), and (ii) the properties related to the protocol itself, such as guaranteed termination and convergence. In doing so, we plan to build on results from literature, such as [14,15] for (i), and [16] for (ii).

This work represents a proposal for a framework to tackle issues that we consider important in multi-source knowledge-intensive applications, and to the best of our knowledge its approach is original. In the future, we intend to complete the formalisation of the aspects that are not already covered by our model. For instance, it seems interesting to work on the definition of further revision operators and on giving a declarative semantics to some specific patterns of interaction, such as the one suggested in Section 5, and, also, to provide a computational counterpart to it.

References

1. SOCS: Societies Of ComputeeS (SOCS): a computational logic model for the description, analysis and verification of global and open societies of heterogeneous computees. `http://lia.deis.unibo.it/research/socs`
2. Kowalski, R.A., Sadri, F.: From logic programming towards multi-agent systems. Annals of Mathematics and Artificial Intelligence **25** (1999) 391–419
3. Leite, J.A., Alferes, J.J., Pereira, L.M.: $\mathcal{MINERVA}$: A dynamic logic programming agent architecture. In: Intelligent Agents VIII: 8th International Workshop, ATAL 2001. LNAI, Vol. 2333. Springer-Verlag (2002)
4. Satoh, K., Inoue, K., Iwanuma, K., Sakama, C.: Speculative computation by abduction under incomplete communication environments. In: Proceedings of the 4th International Conference on Multi-Agent Systems, IEEE Press (2000) 263–270
5. Dell'Acqua, P., Nilsson, U., Pereira, L.M.: A logic based asynchronous multi-agent system. Electronic Notes in Theoretical Computer Science **70(5)** (2002)
6. Sadri, F., Toni, F., Torroni, P.: An abductive logic programming architecture for negotiating agents. In: Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA). LNCS, Vol. 2424. Springer-Verlag (2002) 419–431
7. Ciampolini, A., Lamma, E., Mello, P., Toni, F., Torroni, P.: Co-operation and competition in *ALIAS*: a logic framework for agents that negotiate. Annals of Mathematics and Artificial Intelligence **37** (2003) 65–91
8. Eshghi, K., Kowalski, R.A.: Abduction compared with negation by failure. In: Proc. 6th ICLP, MIT Press (1989) 234–255
9. Dignum, F., Greaves, M., eds.: Issues in Agent Communication. In: Issues in Agent Communication. LNCS, Vol. 1916, Springer-Verlag (2000)
10. Huget, M.P., Dignum, F., eds.: Proceedings of the AAMAS Workshop on Agent Communication Languages and Conversation Policies (2003)
11. Davidsson, P.: Categories of artificial societies. In: Engineering Societies in the Agents World II. LNAI, Vol. 2203. Springer-Verlag (2001) 1–9
12. Dellarocas, C.: The design of reliable trust management systems for online trading communities (2002) Working paper. Available electronically.
13. Alberti, M., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Specification and verification of agent interactions using social integrity constraints. Electronic Notes in Theoretical Computer Science **85** (2003)
14. Eiter, T., Makino, K.: On Computing all Abductive Explanations. In: Proc. AAAI'02, Edmonton, Alberta, Canada. (2002) 62–67
15. Lin, F., You, J.: Abduction in logic programming: a new definition and an abductive procedure based on rewriting. Artificial Intelligence **140** (2002) 175–205
16. Torroni, P.: A study on the termination of negotiation dialogues. In: Proc. AAMAS-2002, Part III, ACM Press (2002) 1223–1230