

# A New Framework for Knowledge Revision of Abductive Agents Through Their Interaction

Andrea Bracciali<sup>1</sup> and Paolo Torroni<sup>2</sup>

<sup>1</sup> Dipartimento di Informatica, Università di Pisa,  
Via Buonarroti, 2 - 56127 Pisa, Italy  
[braccia@di.unipi.it](mailto:braccia@di.unipi.it)

<sup>2</sup> DEIS, Università di Bologna,  
Viale Risorgimento, 2 - 40136 Bologna, Italy  
[paolo.torroni@unibo.it](mailto:paolo.torroni@unibo.it)

**Abstract.** In this paper we discuss the design of a knowledge revision framework for abductive reasoning agents, based on interaction. This involves issues such as: how to exploit knowledge multiplicity to find solutions to problems that agents may not individually solve, what information must be passed or requested, how agents can take advantage of the answers that they obtain, and how they can revise their reasoning process as a consequence of interacting with each other. We describe a novel negotiation framework in which agents will be able to exchange not only abductive hypotheses but also meta-knowledge, which, in particular in this paper, is understood as agents' integrity constraints. 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), Business Process Management (Agentis Software), the Oracle Intelligent Agents, not to mention decentralized control and scheduling, and e-procurement (Rock-

well Automation, Whitestein Technologies and formerly Living Systems AG, Lost Wax, iSOCO), just to cite some.<sup>1</sup>

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. This reason has recently motivated several Logic Programming based approaches to Multi-Agent Systems. Work done by Kowalski and Sadri [1] on the agent cycle, by Leite et al. [2] on combining several Non-Monotonic Reasoning mechanisms in agents, by Satoh et al. [3] on speculative computation, by Dell'Acqua et al. [4, 5] on agent communication and updates, by Sadri et al. [6] on agent dialogues, and by Ciampolini et al. [7] and by Gavanelli et al. [8] 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 [3] 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. In [4] and previous work, the 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, knowledge exchange, update and revision play a key role. We argue that agents would benefit from the capability of exchanging information in its various forms, like predicates, theories and integrity constraints, and that they should be able to do it as a result of a negotiation process regarding knowledge itself.

Finally, the relevance of abduction in proposing revisions during theory refinements and updates is widely recognised. In [9], for instance, it is shown how to combine abduction with inductive techniques in order to “adapt” a knowledge base to a set of empirical data and examples. Abductive explanations are used to generalise or specify the rules of the knowledge base according to positive and negative information provided. In agreement with the basis of this approach, we discuss knowledge sharing mechanisms for agents based on an abductive framework, where information is provided by the interaction between agents, which exchange not only facts, but also constraints about their knowledge, namely in-

---

<sup>1</sup> A summary about industrial applications of agent technology, including references to the above mentioned projects and applications, can be found at: <http://lia.deis.unibo.it/~pt/misc/AIIA03-review.pdf>.

tegrity constraints of an abductive theory. Agent knowledge bases can hence be generalised or specialised by relaxing or tightening their integrity constraints.

In this paper we focus on information exchange about integrity constraints, abstracting away from issues such as ontology and communication languages and protocols: we assume that all agents have a common ontology and communicate by using the same language. In this scenario, autonomous agents will actively ask for knowledge, e.g. facts, hypotheses or integrity constraints, and will autonomously 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, and possibly decide to relax its own constraints in a way that allows him to explain the observation. Conversely, an agent may find out that some assumptions that he made ought to be inconsistent (for instance, due to “social constraints” [10]), and try to gather information about how to tighten his own constraints or add new ones which prevent him from making such assumptions.

The distinguishing features of the distributed reasoning revision paradigm 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. The following example informally illustrates such a scenario.

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

(1) $A \not\models_{\mathcal{IC}''} \neg f, b$	Agent $A$ is unable to prove (find an explanation for) the goal (observation) “there is a <i>bird</i> that does not fly”, and he is <i>able to determine</i> a set $\mathcal{IC}''$ of integrity constraints which prevent to explain the observation ...
(2) $A \rightarrow B : \neg f, b$	... hence $A$ <i>asks</i> $B$ for a possible explanation ...
(3) $B \models_{\Delta}^{\mathcal{IC}'} \neg f, b$	... $B$ is able to explain the observation $\neg f, b$ (e.g., by assuming a set of hypotheses $\Delta$ including $p$ : a <i>penguin</i> is a <i>bird</i> that does not fly), and also to <i>determine</i> a ( <i>significant</i> ) set $\mathcal{IC}'$ of integrity constraints involved in the abductive proof;
(4) $B \rightarrow A : \mathcal{IC}'$	$B$ <i>suggests</i> $\mathcal{IC}'$ to $A$ ;
(5) $A_{\ominus \mathcal{IC}'' \oplus \mathcal{IC}'} \models_{\Delta'} \neg f, b$	$A$ <i>revises</i> his own constraints according to the information provided by $B$ , by means of some <i>updating operation</i> , and explains his observation, possibly with a different explanation $\Delta'$ .

At step (5), the updating operation has been accepted by  $A$ , according to a *conservative policy*, since it *relaxes* the original constraints of  $A$ , and in particular

those preventing to explain the observation ( $\mathcal{IC}''$ ), with a *more generic constraint* provided by  $B$ , e.g.  $\{b, \neg f, \neg p \rightarrow false\}$  (a bird does not fly, unless it is a penguin).

This example will be further elaborated later on, once the necessary notation is introduced. The various steps involve, to some extent, deduction, introspection, interaction, and revision, and will be further discussed to illustrate the global picture. Then we will focus on the integrity constraint revision process, as part of the overall framework.

The rest of this paper is organised as follows: Section 2 recalls Abductive Logic Programming, Section 3 discusses the above mentioned general steps. Section 4 presents the formal model we devised to address constraint-based knowledge revision: an algebra of constraints, relevant constraint selection operators, and constraint updating/revision operators. Possible applications of the framework in knowledge negotiation scenarios are illustrated in Section 5. Concluding remarks and future work are summarised in Section 6.

## 2 Background on Abductive Logic Programming

An Abductive Logic Program (ALP) is a triple  $\langle T, \mathcal{A}, \mathcal{IC} \rangle$ , where  $T$  is a theory, namely a Logic Program,  $\mathcal{A}$  is the set of “abducible” predicates, and  $\mathcal{IC}$  is a set of integrity constraints. According to [11], 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  $\mathcal{IC}$ . In line with [12], along with the abducible (positive) predicates,  $\mathcal{A}$  will also contain all negated literals, which will be generally left implicit. Given an ALP and a goal  $G$ , or “observation”, *abduction* is the process of determining a set  $\Delta$  of abducibles ( $\Delta \subseteq \mathcal{A}$ ), such that:

$$\begin{aligned} T \cup \Delta &\models G, \text{ and} \\ T \cup \mathcal{IC} \cup \Delta &\not\models \perp. \end{aligned}$$

For the sake of simplicity, we assume that  $T$  is ground and stratified. In this case, many of the semantics usually adopted for abduction, like stable model semantics or well-founded semantics, are known to coincide, and hence the symbol  $\models$  can be interpreted as denoting entailment according to any of them. If there exists such a set  $\Delta$ , we call it an *abductive explanation* for  $G$  in  $\langle T, \mathcal{A}, \mathcal{IC} \rangle$ :

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

Abduction is reasoning in presence of uncertainty, represented as the possibility to assume the truth or falsity of some predicates (called “abducibles”) 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,  $\mathcal{IC}$  restricts the choice of possible explanations to observations, or in other words, it rules out some hypothetical worlds from those modelled by a given ALP.

**Syntax and Semantics of Integrity Constraints.** We consider integrity constraints of the form  $\perp \leftarrow L_1, \dots, L_n$ , where  $L_1, \dots, L_n$  are literals, i.e. atoms or negations of atoms, whose set is indicated as  $body(ic)$ . Singleton integrity constraints are indicated as  $ic, ic_1, ic_2, ic', ic'', \dots$ , sets of integrity constraints as  $\mathcal{IC}, \mathcal{IC}_1, \mathcal{IC}_2, \dots, \mathcal{IC}', \mathcal{IC}'', \dots$ , and sets of literals as  $\Delta, \Delta_1, \Delta_2, \dots, \Delta', \Delta'', \dots$ . We will use capital letters to refer specific agents (i.e.,  $A, \mathcal{IC}_A, T_A, \dots$  in Example 2) and lower case to denote generic knowledge without referring to a specific agent (i.e.,  $a, \mathcal{IC}_a, \mathcal{IC}_b, \dots$  in Section 4.3). Finally, in the following, we will adopt the notation  $\leftarrow L_1, \dots, L_n$  as a shorthand for  $\perp \leftarrow L_1, \dots, L_n$ .

Intuitively, an integrity constraint  $\leftarrow L_1, \dots, L_n$  represents a restriction, preventing  $L_1, \dots, L_n$  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. Constraints of the form  $\perp \leftarrow x, \neg x$ , with  $x$  an atom, which prevent  $x$  and  $\neg x$  from being true at the same time, are left *implicit* in the agents' abductive logic programs, and they can be used to implement negation by default.

*Example 2.* Let us consider agent  $A$  of Example 1. Its abductive program states that something can be considered a bird if it either flies or it has feathers, while it can be considered a dolphin if it swims and has no hair, or a mammal if it has hair:

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

In order to explain its observations, agent  $A$  can make assumptions according to its set  $\mathcal{A}$ , and, for instance, classify an animal as a dolphin by assuming that it is able to swim. Note how abducibles have no definitions.  $\mathcal{IC}_A$ , together with all the implicit integrity constraints, prevents  $A$  from considering in their models a bird that does not fly or a dolphin that does not swim. It is clear that there is no  $\Delta \subseteq \mathcal{A}$  such that  $T_A \cup \Delta \models b, \neg f$  and  $T_A \cup \Delta \cup \mathcal{IC}_A \not\models \perp$ , 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 results, specifically about constraint-based knowledge revision.

### 3.1 Communication

Points (2) and (4) 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 [13, 14], 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 (4) 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] and [8], 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 (5). Moreover, agents could be required to explain an observation, starting from a given set of assumptions, or using them in order to determine sets of integrity constraints that have supported (or denied) a proof yielding those explanations. Similarly, it is also necessary to communicate sets of constraints, point (4), 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 [15], 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 [16], based on some form of reputation or 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 in literature, such as [4] 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 [10] 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 to 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 because that they contribute to defining a set of assumptions in a successful proof, or because they make a proof fail. For instance in point (3),  $B$ , relatively to the explanation  $\Delta$ , determines the set  $\mathcal{IC}'$  of integrity constraints supporting the explanation of the observation.

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  $\mathcal{IC}' = \{\leftarrow b, \neg f, \neg p\}$  which is relevant for the proof of the query, and relaxes (see below) the set  $\mathcal{IC}''$ , points (3) and (5) of the example. The definition of a proof procedure implementing the declarative definition of relevance is one of our ongoing activities, although we reckon that the problem of determining significant integrity constraints for a given proof may be computationally hard.

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 search process for suitable 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 dialogue.

### 3.3 Constraint Hierarchies

Point (5) requires the definition of the concept of *relaxation* of a set of constraints, i.e. the capability to determine, given a set  $\mathcal{IC}$ , another set  $\mathcal{IC}'$  that relaxes it. In order to address this issue, we propose in Section 4.2 an algebra of  $\mathcal{IC}$ s, provided with a relaxation-based partial order.

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 proven with respect to  $\mathcal{IC}'$ .

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 relation is the set inclusion partial order. More in general, the relation may fail to be a partial order, in which case it is a pre-order, and checking whether a set relaxes another one according to the definition of relaxing relation may require a more complex analysis, based on the semantics of the corresponding ALP. This point is currently under investigation.

Clearly, the partial order can also be 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 be proved with respect to  $\mathcal{IC}$ . A possible definition of  $<^{IC}$  can be found in Section 4.2. 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 sections.

### 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 (5), 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,  $\mathcal{IC}' = \mathcal{IC}_a \uplus \mathcal{IC}_b$  which relaxes  $\mathcal{IC}_a$  in case  $\mathcal{IC}_a <^{IC} \mathcal{IC}_b$  (see Section 4.3). We present in this paper a specific operator  $\uplus$ , but indeed, within our framework, other tightening and revising operations can be defined.

*Example 3.* After having explained in more detail the steps for constraint revision, let us now possible reconsider Example 1, and let  $A$ 's program be the one introduced in Example 2.

- |   |   |
|---|---|
| (1) $A \not\vdash^{abd} \neg f, b$                    | $A$ is not able to prove $\neg f, b$ and $\mathcal{IC}'' = \{\leftarrow b, \neg f\}$ is not satisfied by any possible explanation for the observation, and hence are candidate for a possible revision; |
| (2) $A \rightarrow B : \neg f, b$                     | $A$ asks $B$ about its observation;   |
| (3) $B \vdash_{\Delta \mathcal{IC}'}^{abd} \neg f, b$ | $\Delta = \{\neg f, p, \neg ha\}$ is an abductive explanation of $B$ , and $\mathcal{IC}' = \{\leftarrow b, \neg f, \neg p\}$ is a relevant constraint for the proof of $B$ ;                           |

- (4)  $B \rightarrow A : \mathcal{IC}'$   $\left\| \begin{array}{l} B \text{ suggests to } A \text{ the (singleton) set of integrity constraints } \mathcal{IC}' = \{\leftarrow b, \neg f, \neg p\}; \end{array} \right.$
- (5)  $A_{\oplus \mathcal{IC}'' \oplus \mathcal{IC}'} \stackrel{abd}{\models}_{\Delta'} \neg f, b$   $\left\| \begin{array}{l} A \text{ relaxes its constraints using } \mathcal{IC}' \text{ and is now able to explain his original observation } \neg f, b, \text{ by using a } \Delta' = \{\neg f, p\} \end{array} \right.$

## 4 A Formal Model for Constraint Revision

In this section we illustrate some technical aspects we devised in order to support the model for constraint revision that we informally discussed in the previous sections. First, we define the set of integrity constraints that are significant in a successful or failed proof; second, 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 below characterises a set of the constraints involved in a successful abductive proof, relatively to a (minimal) set of explanations  $\Delta$ . 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 assumptions can be produced by a successful proof for the same goal.

**Definition 1.** Let  $P = \langle T, \mathcal{A}, \mathcal{IC} \rangle$  be an abductive logic program and  $G$  a goal, such that  $\langle T, \mathcal{A}, \mathcal{IC} \rangle \stackrel{abd}{\models}_{\Delta} G$  (i.e.,  $\Delta$  is a possible explanation for  $G$  in  $P$ ). Let us assume that  $\Delta$  is minimal, i.e.  $\nexists \hat{\Delta} \subset \Delta$  such that  $\langle T, \mathcal{A}, \mathcal{IC} \rangle \stackrel{abd}{\models}_{\hat{\Delta}} G$ .<sup>2</sup> A set of integrity constraints  $\mathcal{IC}' \subseteq \mathcal{IC}$  is relevant with respect to  $P, G$  and  $\Delta$ , written  $\mathcal{IC}' \text{ rel } (\langle T, \mathcal{A}, \mathcal{IC} \rangle, G, \Delta)$ ,  $\stackrel{def}{\iff}$

$$\forall ic \in \mathcal{IC}' \exists \Delta' \subset \Delta \text{ such that } \langle T, \mathcal{A}, \mathcal{IC} \setminus ic \rangle \stackrel{abd}{\not\models}_{\Delta'} G. \quad (i)$$

*Example 4.* Let us consider agent  $B$  of Example 1. Its program  $T_B$  says that something can be considered a bird if either it flies or it is a penguin, or in any case if it is not a mammal. Something is a mammal if it has hair. Its constraints in  $\mathcal{IC}_B$  say that a bird either flies or is a penguin, and that dolphins swim.

$$T_B = \left\{ \begin{array}{l} b \leftarrow f. \\ b \leftarrow p. \\ b \leftarrow \neg m. \\ m \leftarrow ha. \end{array} \right\} \quad \mathcal{A} = \{f, p, ha\} \quad \mathcal{IC}_B = \left\{ \begin{array}{l} \leftarrow b, \neg f, \neg p. \\ \leftarrow d, \neg s. \end{array} \right\}$$

<sup>2</sup> One amongst the minimal sets can be freely chosen.

Agent  $B$  can assume that something flies or it is a penguin, or it has hair, according to its set  $\mathcal{A}$ .

It turns out that  $\langle T_B, \mathcal{A}, \mathcal{IC}_B \rangle \stackrel{abd}{\models}_{\{\neg f, p, \neg ha\}} \neg f, b$ , and  $\{\neg f, p, \neg ha\}$  is minimal. Moreover,

$$\mathcal{IC}' = \{\leftarrow b, \neg f, \neg p\} \mathbf{rel} (\langle T_B, \mathcal{A}, \mathcal{IC}_B \rangle, \neg f, b, \{\neg f, p, \neg ha\}),$$

since  $\langle T_B, \mathcal{A}, \mathcal{IC}_B \setminus \mathcal{IC}' \rangle \stackrel{abd}{\models}_{\{\neg f, \neg ha\}} \neg f, b$ , and  $\{\neg f, \neg ha\} \subset \{\neg f, p, \neg ha\}$ .

In the following, we will use the simplified notation  $\mathcal{IC}' \mathbf{rel} (\langle T, \mathcal{A}, \mathcal{IC} \rangle, G)$  meaning  $\exists \Delta \mid \mathcal{IC}' \mathbf{rel} (\langle T, \mathcal{A}, \mathcal{IC} \rangle, G, \Delta)$

The previous example shows how Definition 1 characterises the set of constraints that  $B$  is able to propose to  $A$  in points (3) and (4) of our Example 1. The definition represents a “sufficient” condition to individuate relevant constraints. Even if more general criteria are under investigation, this simple definition allows us to substantiate the approach. We are also studying its computational counterpart.

On the other hand, given a failing derivation  $\langle T, \mathcal{A}, \mathcal{IC} \rangle \not\stackrel{abd}{\models} G$  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.

**Definition 2.** Let  $P = \langle T, \mathcal{A}, \mathcal{IC} \rangle$  be an abductive logic program, and  $G$  a goal such that  $\langle T, \mathcal{A}, \mathcal{IC} \rangle \not\stackrel{abd}{\models} G$  (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  $\mathcal{IC}' \mathbf{min\_relax} (\langle T, \mathcal{A}, \mathcal{IC} \rangle, G)$ ,  $\stackrel{def}{\iff}$

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

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

*Example 5.* Let us consider an agent  $A$  whose abductive logic program is the one introduced in Example 2. Trivially, it turns out that

$$\mathcal{IC}' = \{\leftarrow b, \neg f\} \mathbf{min\_relax} (\langle T_A, \mathcal{A}, \mathcal{IC}_A \rangle, b, \neg f).$$

## 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 be 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 to define the revision operators.

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

$$ic_i <^{IC} ic_j \stackrel{def}{\iff} \forall \Delta, \Delta \cup \{ic_i\} \not\models \perp \Rightarrow \Delta \cup \{ic_j\} \not\models \perp.$$

*Example 6.* Let us consider the following constraints:

$$\begin{aligned} (ic_1) \quad & \perp \leftarrow a, b, \\ (ic_2) \quad & \perp \leftarrow a, b, c, \end{aligned}$$

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 constraints,  $\mathcal{IC}_i$  and  $\mathcal{IC}_j$ ,

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

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  $ic_j$ ,

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

$$\{ic_i\} <^{IC} \{ic_j\} \Leftrightarrow ic_i <^{IC} ic_j \tag{v}$$

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

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

*Example 8.* Let us consider the following constraints:

$$\begin{aligned} (ic_1) \quad & \perp \leftarrow a, b, \\ (ic_3) \quad & \perp \leftarrow c, \neg b, \\ (ic_4) \quad & \perp \leftarrow a, c, \text{ and the implicit constraint} \\ (ic_5) \quad & \perp \leftarrow b, \neg b. \end{aligned}$$

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

### 4.3 A Constraint Updating Operator

We define an “update with relaxation” operator  $\mathcal{IC}_a \uplus \mathcal{IC}_b$ , 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  $ic_a \notin \mathcal{IC}_a \uplus \mathcal{IC}_b$ . Example 9 illustrates the use of the operator, whose definition is straightforward for the basic cases 1. and 2. of the next definition. Further examples of its usage in Section 5 will hopefully make the rationale of its

definition clearer, in particular when the set  $\mathcal{IC}_b$  has been proposed by a collaborative agent as a relaxation for explaining some facts. This operator supports the revision step done by  $A$  in point (5) of the Example 1.

**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  $\mathcal{IC}_a \uplus \mathcal{IC}_b$ , is defined as follows:

$$\mathcal{IC}_a \uplus \mathcal{IC}_b \stackrel{def}{\iff} \mathcal{IC}_a \setminus \mathcal{IC}'_a \cup \mathcal{IC}_b,$$

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

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

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

1. given two integrity constraints,  $ic_a$  and  $ic_b$ ,

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

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

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

*Example 9.*

Given the following constraints:      The following propositions hold:

$$\begin{array}{ll} (ic_1) & \leftarrow a, b. & (p_1) & ic_1 <^{IC} ic_6; \\ (ic_3) & \leftarrow c, \neg b. & (p_2) & \{ic_1, ic_3, ic_5\} <_{(min)}^{IC} ic_4; \\ (ic_4) & \leftarrow a, c. & (p_3) & \{ic_6\} = \{ic_1\} \uplus \{ic_6\}; \\ (ic_5) & \leftarrow b, \neg b. & (p_4) & \{ic_1, ic_6\} = \{ic_6\} \uplus \{ic_1\}; \\ (ic_6) & \leftarrow a, b, \neg c. & (p_5) & \{ic_4\} = \{ic_1, ic_3, ic_5\} \uplus \{ic_4\}; \\ & & (p_6) & \{ic_3, ic_4, ic_6\} = \{ic_1, ic_3\} \uplus \{ic_4, ic_6\}; \\ & & (p_7) & \{ic_4, ic_5\} = \{ic_1, ic_3, ic_5\} \uplus \{ic_4, ic_5\}; \end{array}$$

## 5 An Example of Application of Our Framework

In this section we show a possible instantiation of the general framework. 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 (pro-actively) 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 reacting 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 **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.

In **Cycle 1**, 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 $a$

---

To explain an observation  $G$ , if  $\langle T_a, \mathcal{A}, \mathcal{IC}_a \rangle \stackrel{abd}{\not\models} G$ :

- 1: Find a minimal relaxation,  $\mathcal{IC}'_a$  **min\_relax** ( $\langle T_a, \mathcal{A}, \mathcal{IC}_a \rangle, G$ )
  - 2: **repeat**
  - 3:   Send **request**( $a, b, \mathcal{IC}'_a, G$ )
  - 4:   Wait for **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**  $\langle T_a, \mathcal{A}, \mathcal{IC}''_a \rangle \stackrel{abd}{\models}_\Delta G$  **then**
  - 8:       Update  $\mathcal{IC}_a$  by  $\mathcal{IC}''_a$
  - 9:     **else**
  - 10:       Send **request**( $a, b, \mathcal{IC}'_a, G$ )
  - 11:       Wait for **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**  $\langle T_a, \mathcal{A}, \mathcal{IC}'_a \rangle \stackrel{abd}{\models}_\Delta G$  or  $\mathcal{IC}'_b = \emptyset$
  - 15:   **if**  $\langle T_a, \mathcal{A}, \mathcal{IC}_a \rangle \stackrel{abd}{\not\models} G$  **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**  $\langle T_a, \mathcal{A}, \mathcal{IC}_a \rangle \stackrel{abd}{\models}_\Delta G$  or  $\mathcal{IC}^1_a = \emptyset$
-

The intuitive reading of **Cycle 1** 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  $G$  (**1:**);
2. ask  $b$  (**3:**) whether he can tell him something about it (more details about  $b$ 's reply are given below);
3. once  $a$  receives a **reply** from  $b$  (**4:**), **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 (**8:**) 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 (**14:**) to communicate to him;
5. if  $a$  does not succeed with  $\mathcal{IC}'_a$  (**15:**), the cycle is repeated until either  $a$  has managed to explain his observation, or there are no more “new” possible minimal relaxations (**19:**).

A dialogue initiated by an agent by means of **request**( $a, b, \mathcal{IC}'_a, G$ ) terminates when the cycle is terminated. The use of such a cycle to try and achieve an agent's needs 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. We will comment more on this in the following.

## 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. check if he can find an abductive explanation for  $G$  (**1:**), and answer back with an empty set in case of failure (**2:**);
2. otherwise single out a minimal subset  $\mathcal{IC}'_b$  of integrity constraints (**4:**) 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  $\mathcal{IC}'_b$  to  $a$  (**7:**): **reply**( $b, a, \mathcal{IC}'_a, G, \mathcal{IC}'_b$ )

This policy is only one example. Other ones are indeed possible. A reason why we have chosen the combination of **Policy 1** and **Cycle 1** is that they allow for exchanging meta-knowledge (constraints) by way of dialogues of the kind of the one shown in the introduction (see Example 10 below). Another reason is that, under some assumptions (atomicity), this combination guarantees termination of the interaction process. In fact, let us consider two agents,  $a$  and  $b$ , each

---

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

---

To reply to a **request**( $a, b, \mathcal{IC}'_a, G$ ):

- 1: **if**  $\langle T_b, \mathcal{A}, \mathcal{IC}_b \rangle \stackrel{abd}{\neq} G$  **then**
  - 2:  $\mathcal{IC}'_b = \emptyset$
  - 3: **else**
  - 4:  $\mathcal{IC}'_b$  is a subset of  $\mathcal{IC}_b$  with the following properties:
    - (i)  $\mathcal{IC}'_b$  **rel** ( $\langle T_b, \mathcal{A}, \mathcal{IC}_b \rangle, G$ );
    - (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);
    - (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 the same  $\mathcal{IC}'_b$  twice in reply to the same request of  $a$ 's);
  - 5: if such a subset cannot be found, then  $\mathcal{IC}'_b = \emptyset$
  - 6: **end if**
  - 7: **reply**( $b, a, \mathcal{IC}'_a, G, \mathcal{IC}'_b$ )
- 

of them behaving according to **Cycle 1** and **Policy 1** when they are engaged in a dialogue. Let us assume that the interaction happens “atomically”, i.e., once  $a$  initiates a dialogue by a **request** to explain a goal  $G$ , neither agent will update his knowledge base until  $a$  succeeds in explaining  $G$ , or the interaction has terminated. Then, an interaction protocol initiated by either agent will terminate in a finite number of steps. Informally, this is the case because, given a set  $\mathcal{IC}$ , there is a finite number of subsets of  $\mathcal{IC}$ ;<sup>3</sup> at each iteration a different subset of  $\mathcal{IC}$  is considered, and the two agents will not revise their knowledge base until the interaction has terminated, either successfully or unsuccessfully. Since no message is exchanged twice, and since there is a finite number of messages that  $a$  and  $b$  can exchange, the interaction finishes in a finite number of steps.

Another property that we would like to achieve 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.

A last remark, before we conclude by giving an example where two agents interact by following **Cycle 1** and **Policy 1**. 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 prevented by any constraint of  $a$ 's, but instead  $a$  is unable to explain  $G$  only because  $a$  is missing some definition which  $b$  instead knows. In order to tackle this situation, we will need to consider different introspection and updating operators, which need to reason upon and modify not only the

---

<sup>3</sup> We are considering in this work only propositional abductive logic programs. If we extend it to the non-propositional case, the cycle/policy would also have to be suitably adapted.

agents' integrity constraints, but also their programs, and we would need to find some suitable cycles and policies that support such a process.

*Example 10.* Let us consider two agents  $A$  and  $B$ , where the abductive logic program of  $A$  is that defined in Example 2, and the abductive logic program of  $B$  is that defined in Example 4, as reported below. We assume now that  $A$  and  $B$  have the same set of abducible predicates  $\mathcal{A} = \{f, s, fe, ha, p\}$ .

$$T_A = \left\{ \begin{array}{l} b \leftarrow fe. \\ b \leftarrow f. \\ d \leftarrow s, \neg ha. \\ m \leftarrow ha. \end{array} \right\} \quad \mathcal{IC}_A = \left\{ \begin{array}{l} (ic_1) \leftarrow b, \neg f. \\ \leftarrow d, \neg s. \end{array} \right\}$$

$$T_B = \left\{ \begin{array}{l} b \leftarrow f. \\ b \leftarrow p. \\ b \leftarrow \neg m. \\ m \leftarrow ha. \end{array} \right\} \quad \mathcal{IC}_B = \left\{ \begin{array}{l} (ic_2) \leftarrow b, \neg f, \neg p. \\ \leftarrow d, \neg s. \end{array} \right\}$$

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  $G = \{\neg f, b\}$ . Then, the behaviour of the system  $\{A, B\}$  is the following:

- (1)  $\langle T_A, \mathcal{A}, \mathcal{IC}_A \rangle \stackrel{abd}{\not\models} \{\neg f, b\}$ . At this point  $A$  looks for a minimal relaxation of its constraints towards explaining  $\{\neg f, b\}$  (see Example 5), and it finds it in  $\{ic_1\} = \{\leftarrow b, \neg f.\}$ :

$$\{\leftarrow b, \neg f.\} \text{ min\_relax } (\langle T_A, \mathcal{A}, \mathcal{IC}_A \rangle, G)$$

- (2) Following **Cycle 1**,  $A$  asks  $B$  a set of relevant integrity constraints of his, that are relevant for the explanation of  $G$ :

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

- (3) Upon receipt of  $A$ 's request, according to **Policy 1**  $B$  verifies that he is able to prove  $G$ :  $\langle T_B, \mathcal{A}, \mathcal{IC}_B \rangle \stackrel{abd}{\models}_{\{\neg f, p, \neg ha\}} \{b, \neg f\}$ .  $B$  finds a set of integrity constraints  $\mathcal{IC}'_B = \{ic_2\}$  that are relevant to  $A$ 's request (see Example 4):

$$\{\leftarrow b, \neg f, \neg p.\} \text{ rel } (\langle T_B, \mathcal{A}, \mathcal{IC}_B \rangle, \{b, \neg f\}),$$

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

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

- (4)  $A$  generates the set  $\mathcal{IC}''_A = \left\{ \begin{array}{l} \leftarrow b, \neg f, \neg p. \\ \leftarrow d, \neg s. \end{array} \right\}$  as a revision of its own  $\mathcal{IC}_A$  by

$\mathcal{IC}'_B$ :  $\mathcal{IC}''_A = \mathcal{IC}_A \uplus \mathcal{IC}'_B$ . Since  $\langle T_A, \mathcal{A}, \mathcal{IC}''_A \rangle \stackrel{abd}{\models}_{\{b, \neg f, p\}} \{b, \neg f\}$ ,  $A$  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 [17], 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 to be inconsistent. Exchanging constraints help revising the reasoning capabilities of agents. 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. In the last section of this paper, we showed a possible application of the framework to specific interaction patterns, and we commented 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 [18, 19] for *(i)*, and [20] 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.

## Acknowledgments

We would like to thank Luís Pereira and the anonymous referees for their valuable comments and feedback. This work was partially funded by the Information Society Technologies programme of the European Commission, Future and Emerging Technologies under the IST-2001-32530 SOCS project [17], within the Global Computing proactive initiative, and by the MIUR COFIN 40% project *Sviluppo e verifica di sistemi multiagente basati sulla logica*.

## References

1. Kowalski, R.A., Sadri, F.: From logic programming towards multi-agent systems. *Annals of Mathematics and Artificial Intelligence* **25** (1999) 391–419
2. Leite, J.A., Alferes, J.J., Pereira, L.M.: *MZNERVA*: A dynamic logic programming agent architecture. In: *Intelligent Agents VIII: 8th International Workshop, ATAL 2001, Seattle, WA, USA, Revised Papers*. Lecture Notes in Artificial Intelligence **2333**, Springer-Verlag (2002) 141–157
3. 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, Boston, USA, IEEE Press (2000)* 263–270
4. Dell’Acqua, P., Nilsson, U., Pereira, L.M.: A logic based asynchronous multi-agent system. *Electronic Notes in Theoretical Computer Science* **70** (2002)
5. Dell’Acqua, P.: Weighted multi dimensional logic programs. In this volume.
6. Sadri, F., Toni, F., Torroni, P.: An abductive logic programming architecture for negotiating agents. In Greco, S., Leone, N., eds.: *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA)*. Lecture Notes in Artificial Intelligence **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. *Computational Logic in Multi-Agent Systems*. *Annals of Mathematics and Artificial Intelligence* **37** (2003) 65–91
8. Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: An abductive framework for information exchange in multi-agent systems. In this volume.
9. Mooney, R.J.: Integrating abduction and induction in machine learning. In Flach, P.A., Kakas, A.C., eds.: *Abductive and Inductive Reasoning*. Pure and Applied Logic. Kluwer Academic Publishers (2000) 181–191
10. 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)
11. Eshghi, K., Kowalski, R.A.: Abduction compared with negation by failure. In Levi, G., Martelli, M., eds.: *Proceedings of the 6th International Conference on Logic Programming*, MIT Press (1989) 234–255
12. Kakas, A.C., Mancarella, P.: Generalized stable models: a semantics for abduction. In: *Proceedings 9th European Conference on Artificial Intelligence*, Pitman Publishing (1990)
13. Dignum, F., Greaves, M., eds.: *Issues in Agent Communication*. Lecture Notes in Artificial Intelligence **1916**, Springer-Verlag (2000)

14. Huget, M.P., Dignum, F., eds.: *Advances in Agent Communication: International Workshop on Agent Communication Languages, ACL 2003, Melbourne, Australia, July 14, 2003. Revised and Invited Papers. Lecture Notes in Artificial Intelligence 2922*, Springer-Verlag (2004)
15. Davidsson, P.: *Categories of artificial societies*. In Omicini, A., Petta, P., Tolksdorf, R., eds.: *Engineering Societies in the Agents World II. Volume 2203 of Lecture Notes in Artificial Intelligence.*, Springer-Verlag (2001) 1–9
16. Castelfranchi, C., Falcone, R., Firozabadi, B., Tan, Y.H.:, guest eds.: *Special Issue on “Trust in Agents”, Parts 1 and 2. Applied Artificial Intelligence 14 (2000)*
17. *Societies Of Computees (SOCS): a computational logic model for the description, analysis and verification of global and open societies of heterogeneous computees. Home Page: <http://lia.deis.unibo.it/Research/SOCS/>.*
18. Eiter, T., Makino, K.: *On Computing all Abductive Explanations*. In: *Proceedings of the 18th National Conference on Artificial Intelligence, AAAI’02, Edmonton, Alberta, Canada (2002)* 62–67
19. Lin, F., You, J.: *Abduction in logic programming: a new definition and an abductive procedure based on rewriting. Artificial Intelligence 140 (2002)* 175–205
20. Torroni, P.: *A study on the termination of negotiation dialogues*. In Castelfranchi, C., Lewis Johnson, W., eds.: *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002), Part III, Bologna, Italy, ACM Press (2002)* 1223–1230