An Abductive Framework for Information Exchange in Multi-Agent systems *

Marco Gavanelli[1], Evelina Lamma[1], Paola Mello[2], and Paolo Torroni[2]

[1] Dip. di Ingegneria - Università di Ferrara - Via Saragat, 1 - 44100 Ferrara, Italy.
{mgavanelli|elamma}@ing.unife.it
[2] DEIS - Università di Bologna - Viale Risorgimento, 2 - 40136 Bologna, Italy.
{pmello|ptorroni}@deis.unibo.it

Abstract. In this paper, we propose a framework for information exchange among abductive agents whose local knowledge bases are enlarged with a set of abduced hypotheses. We integrate the aspects of information exchange and abductive reasoning, and show theoretically the information inferred by the single abductive agent as a product of joint reasoning activity. We show examples, like dining philosophers, resource exchange and speculative computation, and give an implementation of the space of interactions based on $CLP(\mathcal{SET})$.

1 Introduction

In the past years, Computational Logics has proved itself to be a powerful tool for modelling and implementing many forms of reasoning of intelligent systems, such as deduction (the basic logic reasoning method), abduction (Abductive Logic Programming) for reasoning from effects to causes, machine learning (Inductive Logic Programming).

Traditionally, such techniques have been developed for monolithic systems, to solve problems such as diagnosis (expert systems) and learning by examples. More recently, following the development of multi-agent research, considerable effort has been done in exporting the technological achievements of Computational Logics into a multi-agent setting [1]. For instance, through abduction, an agent can make hypotheses on the outer world, and on causes of observable events, which is a natural extension of what happens in an expert system. But the role of abduction in multi-agent systems can go beyond the internal agent reasoning.

In general, sociable agents will exchange knowledge, ask questions, provide services to each other and, eventually, get to reasonable agreements when decisions are necessary. In this paper, we propose a Computational Logic-based framework for the integration of abductive reasoning and communication.

One of the first approaches to modelling agents based on Computational Logics was proposed by Kowalski and Sadri [2]. The authors propose an agent cycle where logic agents reason based on an abductive logic program, the hypotheses produced within the agent represent actions in the outer world, and the observations from the outer world are mapped into "abduced" that enlarge the agent's knowledge base. Communication

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

primitives are considered as a particular case of actions. Agent communication can take place for various reasons, e.g., in reaction to stimuli, or as part of a plan to achieve a goal. The sequence of communicative actions can follow a certain protocol, or else they can be less constrained like in the case of agent dialogues. In this setting, the communication plan is based on the abductive reasoning of the agents. The communication primitives (e.g., in/out à la Linda, or higher level primitives, like request, promise, etc.) are explicitly defined. The integrity constraints and the knowledge base that constitute the agent's abductive logic program define the communication protocol by stating explicitly how to react to incoming communication actions. Building on [2], several other proposals have been published that map agent communication acts into abductive atoms [3,4].

In the present work, we take a different approach. By means of the well-understood declarative semantics of abduction, we give a formal understanding of information exchange. In fact, we give an abductive semantics to the whole group of agents, that together, by using their own knowledge bases, will derive the goal. Our approach can be considered top-down, meant to define the semantics of a group reasoning activity, while many approaches are more bottom-up, in the sense that they give a declarative reading of the agents' knowledge bases, and derive a behavior which is hopefully sound to the semantics. The two essential concepts of reasoning and information exchange are nicely integrated in a uniform semantic characterization. We identify the information exchanged when agents come to a global agreement upon hypotheses. Drawing inspiration from ALIAS [5], we group abductive agents interacting together and provide them with a shared repository of communicative actions, which we call Δ. Based on it, we establish abduction as a virtual machine given by communicating agents. All the agents will share the same Δ and a solution must satisfy all the (local) integrity constraints. In this way, communication primitives are transparent to the abductive reasoners and the communication protocol is implicitly defined in the program and in the integrity constraints of the agents. In a sense, we abstract away from protocols: the agents do not need to name explicitly the others in the group, they do not even need to know how many agents participate in the distributed computation, while trying and find a global agreement.

Consider, for example, a system that monitors some electronic equipment. We may have an abductive agent that monitors each of the alarms. A first agent, C_1, is responsible for checking the temperature of the whole system. It may have rules saying that the temperature can get high if the fan in one of the subsystems is broken, or in case of short circuit in one subsystem:

$$KB_1 \qquad \begin{array}{l} high\_temp \leftarrow broken\_device(\mathit{fan}, System). \\ high\_temp \leftarrow short\_circuit(System). \end{array}$$

A second abductive agent, C_2, checks the output of each subsystem, and knows that the wrong output of a system can be due to a failure in a device of such a system.

$$KB_2 \qquad wrong\_output(System) \leftarrow broken\_device(Device, System).$$

Finally, a third agent, C_3, checks the current absorption, and may have an integrity constraint saying that there cannot be, in a subsystem, low current and short-circuit:

$$IC_3 \qquad \leftarrow low\_current(System), short\_circuit(System).$$

Now, if we have high temperature, low current in all subsystems, and wrong output on subsystem *amplifier*2, none of the single agents can get to the right solution; however, together they can identify the failing device (namely, the *fan* in *amplifier*2). Notice that they will need to unify their possible guesses (C_1 could hypothesize $broken\_device(\textit{fan}, S)$ and C_2 $broken\_device(D, \textit{amplifier}2)$) to get to the complete solution. Moreover, they will need to check the integrity constraints of all the abductive agents to get a consistent solution. We see by this simple example how variable binding can be considered as both the subject and the vehicle of communication.

In this paper, we propose a framework of abductive reasoners that share a common hypothesis space. We do not explicitly address some typical issues of Multi-Agent Systems, like autonomy and pro-activity. Our focus is rather on the communication part and its relation with the agent's reasoning. Notice that it is not necessary that each agent exports its whole hypothesis space, but only the part related to communication, while its internal reasoning may remain private. We formally define the framework and theoretically show how the information is passed (Sect. 3).

In this work, we also discuss about the outcome of the collaborative reasoning process in terms of local vs. global consistency. To this end, we introduce the concepts of independence and compositionality of programs, and we prove that, in terms of consistency and under certain conditions, collaborative reasoning is equivalent to local abductive reasoning.

We show various examples of communication patterns that can be obtained in our framework (Sect. 4). We provide a prototypical implementation of our framework (Sect. 5), we discuss related work in Sect. 6 and, finally, we conclude.

2 Preliminaries on Abductive Logic Programming (ALP)

If F is a formula, with $\tilde{\exists}F$ (resp. $\tilde{\forall}F$) we denote the existential (resp. universal) closure of the formula. If t is a term, with $\tilde{\exists}_t F$ (resp. $\tilde{\forall}_t F$) we will indicate the existential (universal) closure of F restricted to the variables in t.

Definition 1. *An* abductive logic program *is a triple $\langle KB, Ab, IC\rangle$ where:*

- *KB is a (normal) logic program, that is, a set of clauses ("definitions") of the form $A_0 \leftarrow A_1, \ldots, A_m$, where each A_i ($i = 1, \ldots, m$) is a positive or negative literal;*
- *Ab a set of* abducible predicates, *p, such that p does not occur in the head of any clause of KB;*
- *IC is a set of integrity constraints, that is, a set of closed formulae.*

Following Eshghi and Kowalski [6], an abductive logic program $\langle KB, Ab, IC\rangle$ can be transformed into its *positive version*. The idea is to view default literals as new abducible positive atoms. In the rest of the paper, we will use the symbol `not` to indicate negation, and suppose that it is treated as by Eshghi and Kowalski [6].

Definition 2. *Given an abductive program $\langle KB, Ab, IC\rangle$ and a goal G, an* abductive explanation *for G is a set Δ (such that $\Delta \subseteq Ab$) with a substitution θ such that $KB \cup \Delta$ is consistent and*

- $KB \cup \Delta \models \tilde{\forall}(G/\theta)$
- $KB \cup \Delta \models IC$

We suppose that each integrity constraint has the syntax

$$(\bot) \leftarrow A_1, \ldots, A_n.$$

where $A_1, \ldots, A_n$ is a conjunction of atoms. Let I be the implication $A_0 \leftarrow A_1, \ldots, A_m$; we call $head(I)$ the atom A_0 and we denote with $body(I)$ the set $\{A_1, \ldots, A_m\}$.

Given this syntax, the previous definition $KB \cup \Delta \models IC$ is equivalent to saying that the atoms appearing in the body of an integrity constraint cannot be all true in order for the program (with the Δ) to be consistent: $\forall ic \in IC, \exists a \in body(ic)$ s.t. $KB \cup \Delta \not\models a$.

3 Formalization

In this section, we give the formalization of our framework for information sharing.

Let C_i denote an agent, provided with an abductive logic program $\langle KB_i, Ab, IC_i\rangle$.[3] In order to (abductively) prove a goal G_i, C_i will try to find a binding θ_i and a set of abductive hypotheses δ_i such that

$$KB_i \cup \delta_i \models G_i/\theta_i$$

that satisfies all the integrity constraints:

$$KB_i \cup \delta_i \models IC_i.$$

If we allow variables in the set δ_i, then the substitution θ_i will also apply to the set δ_i; all the remaining variables in δ_i/θ_i should be considered existentially quantified[4]

$$\tilde{\exists}_{\delta_i/\theta_i}(KB_i \cup \delta_i/\theta_i \models G_i/\theta_i).$$

Communication between agents will appear as a binding on the set of abduced hypotheses. Given n abductive agents C_i, $i = 1..n$, each providing an answer to a goal:

$$\tilde{\exists}_{\delta_i/\theta_i}[KB_i \cup \delta_i/\theta_i \models G_i/\theta_i]$$

[3] Since in this work we are tackling the problem of information sharing in the context of agents reasoning based on abductive logic programs, from now on we will use – with abuse of notation – the same symbol to denote both the agents and the ALP that they enclose. Indeed, in a more elaborated agent architecture, the abductive logic program will only represent a part of the whole agent. Also, in this simplified setting we consider – without loss of generality – the set Ab to be the same for all the agents in the system.

[4] Other variables which may appear in G_i are considered free, as in the IFF [7].

communication will appear as (one) solution of the following equations:

$$\Delta = \bigcup_k \delta_k$$

$$\tilde{\exists}_{\Delta} \forall k (KB_k \cup \Delta \models IC_k). \tag{1}$$

that can be seen as a $CLP(\mathcal{SET})$problem [8,9].

Definition 3. *The property in Eq. 1 will be referred to as* Global Consistency.

Global consistency is equivalent to abduction performed by a single agent that has the union of the KBs and the union of the ICs. In order to prove this property, we introduce the concept of *independency* and *compositionality* of programs, and show that independency implies compositionality.

Definition 4. *A set of atoms Δ is* independent *of a logic program KB, and we write $ind(\Delta, KB)$, iff $\forall a \in \Delta \nexists d \in KB$ s.t. $a/head(d)$.*

Definition 5. *A logic program KB_1 is* independent *of a program KB_2, and we write $ind(KB_1, KB_2)$, iff $\forall d_i \in KB_1$ and $\forall d_j \in KB_2$, $ind(\{head(d_i)\} \cup body(d_i), d_j)$.*

Note that this definition is not symmetric: $ind(KB_1, KB_2) \not\Rightarrow ind(KB_2, KB_1)$.

Theorem 1. Compositionality. *Suppose that KB_1 and KB_2 are mutually independent and that Δ is a set of ground facts independent of KB_1 and KB_2 (but, nevertheless, KB_1 and KB_2 may depend on Δ).*
Then, $\forall a$,

$$KB_1 \cup KB_2 \cup \Delta \models a \quad \Leftrightarrow \quad (KB_1 \cup \Delta \models a) \vee (KB_2 \cup \Delta \models a)$$

Proof *See Appendix.* □

Achieving compositionality in a system of autonomous agents is not difficult: in fact, we can assume that the "private" atoms used by the various agents (those that are not intended for direct sharing) can either have different functor names or arity, or they can be labelled with the name of the agent. Instead, for what regards the abducible predicates used for communication, we can assume [10] that they have no definition.

We now extend Definition 5 for Abductive Logic Programs; intuitively, integrity constraints in one of the programs should not reference predicates defined in the other.

Definition 6. *An* Abductive Logic Program $\langle KB_1, Ab, IC_1 \rangle$ is independent *of a program $\langle KB_2, Ab, IC_2 \rangle$ iff*

- $ind(KB_1, KB_2)$, *and*
- $\forall ic_i \in IC_1, \forall a \in body(ic_i), \nexists d \in KB_2$ *s.d.* $a/head(d)$.

Theorem 2. *Let $\langle KB_1, Ab, IC_1 \rangle$ and $\langle KB_2, Ab, IC_2 \rangle$ be two mutually independent abductive logic programs. Then, global consistency is equivalent to (centralized) abduction, with $KB = \cup_i KB_i$ and $IC = \cup_i IC_i$; i.e., the two following conditions are equivalent*

– $\tilde{\exists}_{\Delta} \forall i (KB_i \cup \Delta \models IC_i)$
– $\cup_i KB_i \cup \tilde{\exists} \Delta \models \cup_i IC_i$

Proof *See Appendix.* □

Note that Global Consistency requires that all the abductive reasoners will "agree" on one substitution of the variables in Δ. A weaker variant is *Local Consistency*:

Definition 7. *A set Δ of abduced hypotheses is* Locally Consistent *if the following condition holds:*

$$\forall i (KB_i \cup \tilde{\exists} \Delta \models IC_i). \tag{2}$$

If each of the agents checks the consistency of the set Δ locally, local consistency is ensured. However, local and global consistency are different properties:

Example 1. Consider the following situation, where $p/2$ is abducible.

$$Agent\ 1 \quad \begin{cases} IC_1 & \leftarrow p(X,Y), not\ q(X,Y). \\ KB_1 & q(X,Y) \leftarrow X > Y. \end{cases}$$

$$Agent\ 2 \quad \begin{cases} IC_2 & \leftarrow p(X,1), not\ f(X). \\ KB_2 & f(X) \leftarrow X < 0. \end{cases}$$

Should both agents try to assume $p(X,Y)$, we could obtain $\Delta = \{p(X,1)\}$, and Agent 1 will receive the binding $Y/1$. This is locally consistent, in fact for Agent 1 there exists a value of X that satisfies its integrity constraint (every value greater than one), and, similarly, for Agent 2 there exists at least one value (any value less than zero) that satisfies IC_2. Obviously, it is not consistent, because there is no value for X that satisfies all the integrity constraints, thus it is not globally consistent, and the hypothesis $(\exists X) p(X,1)$ should be rejected.

One may think, operationally, to enforce only local consistency, because it is less expensive. However, in this case, an eventual inconsistency might not be detected, and have an expensive failure in a later computation.

Various types of communication may appear in this framework, e.g., communication of a failure, communication triggered by integrity constraints, etc. In this paper, we focus on the communication given by a shared abduced hypothesis. Intuitively, when hypotheses made by different agents are unified, communication appears as a binding.

3.1 Communication through a shared abduced hypothesis

Once global consistency is enforced, we can identify the information exchanged among agents, if some of them share (at least) one predicate name in the abducible space. In fact, given two abductive agents, x_i, $i = 1..2$, each enclosing an abductive logic program $\langle KB_i, Ab, IC_i \rangle$, for all i we have:

$$\tilde{\exists}_{\delta_i/\theta_i} KB_i \cup \delta_i/\theta_i \models G_i/\theta_i.$$

If the set inequality $\delta_1/\theta_1 \cap \delta_2/\theta_2 \neq \emptyset$ has solutions, i.e., if there is a substitution θ' such that

$$\exists\theta' : (\delta_1/\theta_1)\theta' \cap (\delta_2/\theta_2)\theta' \neq \emptyset$$

then information exchange can occur by way of a variable binding.

The communication is, in general, bidirectional: both agents will receive the binding θ'. The information that agent 1 will receive is the substitution for its variables, $\theta'|_{\delta_1/\theta_1}$, and, in the same way, agent 2 will receive the information $\theta'|_{\delta_2/\theta_2}$.

Example 2. Let us consider the following instance, where $a/1$ is the only abducible.

	Agent 1		Agent 2
IC_1	$\{\leftarrow a(X), a(Y), X \neq Y.$		
KB_1	$\begin{cases} p(X) \leftarrow a(X), b(X). \\ b(r(1,B)). \end{cases}$	KB_2	$\begin{cases} q(X) \leftarrow a(X), f(X). \\ f(r(A,2)). \end{cases}$

The integrity constraint IC_1 tells that there can be only one atom $a/1$ in the Δ. If the first agent proves $?-p(Z)$ and the second $?-q(Q)$, the individual results will be

$$\theta_1 = \{Z/r(1,B)\} \qquad \delta_1 = \{a(r(1,B))\}$$
$$\theta_2 = \{Q/r(A,2)\} \qquad \delta_2 = \{a(r(A,2))\}$$

In this case there is only one most general unifier, namely $\theta' = \{B/2, A/1\}$. Both of the agents receive this substitution; the received information is the substitution restricted to their variables, i.e., Agent 1 receives $\theta'|_{\delta_1/\theta_1} = \{B/2\}$ and Agent 2 $\theta'|_{\delta_2/\theta_2} = \{A/1\}$.

4 Examples

Various information exchange patterns can be implemented on top of our the framework of abductive communication. In this section, we show some simple examples, that exploit non trivial communication patterns enclosed in the framework.

4.1 Dining Philosophers

The *dining philosophers* problem [11] is a classic problem in inter-process synchronization. The problem consists of a group of philosophers sitting at a table; each philosopher has a chopstick on his right and one on his left. A chopstick cannot be used by two philosophers at the same time, and one philosopher needs two chopsticks to eat.

We do not wish to propose with our framework a new solution to the dining philosophers, but instead, we will use the well-known example to show the inter-process communication involved and the abductive semantics that we give to it.

We propose a model based on abductive agents, each one representing a philosopher; agents share resources (the chopsticks), and communicate by means of abducible predicates that they derive within the knowledge base. Such predicates represent the state of the philosophers with respect to the chopsticks: $chop(C, F, T)$, where C indicates a chopstick, F represents a philosopher (abductive agent) and T is the time.

The set Δ, which grows during the individual computation activities of agents, contains all their abducible predicates, which must be agreed upon by all of them at all times. It represents a partial schedule of the allocation of chopsticks in time. Due to the conflicts on the use of resources (the chopsticks), the reasoning activity must be coordinated, and in particular it must comply with some constraints that must never be violated. For example, a chopstick cannot be taken by two different agents at the same time. In ALP terms, we would impose an integrity constraint such as:

$$\leftarrow \texttt{chop(C,F,T), chop(C,F1,T), F} \neq \texttt{F1} \tag{3}$$

This constraint implies that for all ground instances of the first two predicates, the third one must fail (F must be equal to $F1$). That is, if an agent abduces $chop(1, 1, t)$, then $\forall X, chop(1, X, t) \notin \Delta \setminus \{chop(1, 1, t)\}$.

A resource at a given time is denoted as free by leaving its second variable – the one representing the owner – free. In this way, releasing a resource means abducing a fact with a variable as owner. Acquiring a resource means abducing a `chop` atom in which the second variable is not free. If a chopstick was released at a given time point, then Δ contains $chop(C, F, T)$, where F is a variable. A philosopher $F1$ can take it by abducing $chop(C, F1, T)$, and either F will be unified with $F1$ or (as a choice point) Δ will contain both the abducibles and the constraint $F \neq F1$.

This predicate shows the behavior of a single philosopher:

```
phil(P) ←
 compute_needed_chops(P,Chop1,Chop2),
 chop(Chop1,P,T),  chop(Chop2,P,T),  % Get needed resources
 eat(T,T1),
 chop(Chop1,_,T1), chop(Chop2,_,T1). % Release the resources

compute_needed_chops(P,P,P1) ←
   number_philosophers(Pn), P<Pn, P1 is P+1.
compute_needed_chops(P,1,P) ← number_philosophers(P).
```

Let us see an example with three philosophers: p_1, p_2, and p_3 (all philosophers will have $number\_philosophers(3) \leftarrow$. in their knowledge base). At the beginning (time zero) all the resources are free. We model this by introducing in the Δ three atoms `chops(i,_,0)`, where `i` $= 1..3$. Let us suppose the first philosopher, p_1, tries to get its needed resource first: he will abduce $chop(1, p_1, T1), chop(2, p_1, T1)$, i.e., he will try to get two chopsticks in a same time stamp. Since the Δ contains all the free resources at time zero, the philosopher gets the binding $T1/0$ and the Δ will contain information that two chopsticks are no more available. If philosopher p_2 tries to get its resources, he cannot get them at time zero, because the integrity constraint forbids to abduce both $chop(2, p_1, 0)$ and $chop(2, p_2, 0)$ ($p_1 \neq p_2$); the only possibility is to abduce new facts $chop(2, p_2, T2)$, $chop(3, p_2, T2)$. The second philosopher still does not know in which time tick he will get the resources (T_2 is still a variable). If now p_1 releases its resources at time 3, abducing $chop(2, \_, 3)$, this atom unifies with one request of p_2, so p_2 gets the binding $T2/3$.

start	chop(1,_,0)	chop(2,_,0)	chop(3,_,0)
p_1 get chops	chop(1,p_1,0)	chop(2,p_1,0)	chop(3,_,0)
p_2 ask chops	chop(1,p_1,0)	chop(2,p_1,0)	chop(3,_,0)
		chop(2,p_2,T2)	chop(3,p_2,T2)
p_1 release chops	chop(1,p_1,0)	chop(2,p_1,0)	chop(3,_,0)
	chop(1,_,3)	chop(2,p_2,3)	chop(3,p_2,3)

We specified the program in a general way, i.e., independent of the philosopher. In fact, if we instantiate it to a specific philosopher (e.g., we define $phil(p_1)$ and $compute\_needed\_chops(p_1, P, P1)$ instead of the generic predicates $phil(P)$ and $compute\_needed\_chops(P, P, P1)$ that we defined above) we obtain three mutually independent programs, for which the results of Theorem 2 hold. In that case, global consistency is equivalent to centralized abduction. Similar considerations apply to the other examples that will follow this section.

This example must not be seen as a possible solution to the synchronization problems of the dining philosophers, but rather as an example of information sharing. p_1, p_2, and p_3 will collaborate to generate a schedule of resource usage, and the semantics and properties of the framework ensure that their constraints are not violated. Indeed, different agents with different programs and constraints can participate in the solution of this problem, each one adopting suitable strategies (e.g., coordination mechanisms or ad-hoc ordering to prevent starvation).

4.2 Dialogues and negotiation

In this section we would like to show how it is possible to model in our framework a two-agent dialogue. We will take as an example the negotiation dialogues produced by $\mathcal{N}^+$-systems [4]. Such dialogues – sequences of dialogue moves satisfying certain requirements – can be used to solve a *resource reallocation problem*. $\mathcal{N}^+$-agents produce dialogue moves produced by means of an abductive proof procedure, during the *think* phase of an observe-think-act life cycle [2]. In the context of resource reallocation considered in [4], agents have goals to achieve, resources to use in order to achieve them, and they produce dialogues among each other in order to obtain the resources that they miss to make a certain plan feasible. In the simplified setting that we are considering now, the purpose of producing a dialogue move is either to reply to a request, or to request a missing resource. $\mathcal{N}^+$-agents keep requesting resources to the other agents until they either obtain all the missing resources or they realize that there are not enough resources in the system to make their plan feasible. At the same time, agents must reply to asynchronously incoming requests. The policy used to produce requests and to reply to the other agents' requests is encoded into an abductive logic program.

We propose here an alternative implementation of $\mathcal{N}^+$-agents based on non-ground abducible predicates where an agent does not explicitly poll each other agent in the group, but posts a request in the common Δ with a variable as addressee. Other agents can hypothesize to be the addressee of the message and, consequently, reply.

We express dialogue moves by means of envelopes $\mathtt{t}/3$ that have the following syntax: $\mathtt{t}(\mathtt{Sender}, \mathtt{Receiver}, \mathtt{Subject})$, where `Sender`, `Receiver`, and `Subject` are terms that carry the obvious meaning.

The definition of the predicates is below. We divide the agent resources into two groups: those that are missing (and that the agent must somehow obtain in order to succeed in its goal, `make_plan_feasible`), and those that the agent may give away. For the sake of simplicity, we adopt here a static representation of the problem in which the set of missing resources is defined through a `missing/1` predicate. A resource `r` is instead available if a predicate `available(r)` is true. We consider a setting in which three agents (a, b, and c) have the program below. It is given independently of the agent, but we assume that each agent has its own definition of `missing/1`, `available/1`, and `self/1`, this latter used to provide each agent with a unique identifier.

```
make_plan_feasible ← missing(M), get_all(M).
get_all([]).
get_all([R|R1]) ← get(R), get_all(R1).
get(R) ← self(S),
   t(S,A,request(give(R))), S ≠ A,
   t(A,S,accept(give(R))).
manage_request(S,X,R) ← available(R),
   t(S,X,accept(give(R))).
manage_request(S,X,R) ← not(available(R)),
   t(S,X,refuse(give(R))).
```

The integrity constraints are the following:

```
   ← t(S,R,refuse(give(R))), t(S,R,accept(give(R))).        (4)
← self(S),t(Y,S,request(give(R))),not manage_request(S,Y,R).(5)
```

The first one states that an agent cannot reply both *accept* and *refuse* to the same request. The second one is used to react to a request of other agents by invoking `manage_request` if a request is addressed to the agent.

Let us see an example with three agents, called `a`, `b` and `c`. Suppose that the individual knowledge bases of the three agents are the following:

Agent `a`	Agent `b`	Agent `c`
`self(a).`	`self(b).`	`self(c).`
`missing([nail]).`	`missing([pen]).`	`missing([knife]).`
`available(pen).`	`available(pen).`	`available(nail).`
`available(knife).`		

Suppose that agent `a` starts first, and posts a request for its missing resource: `t(a, X, request(give(nail)))`. It will also try to abduce that the same agent that will reply, X, will accept to give the resource: `t(X, a, request(give(nail)))`. This second hypothesis is motivated by the fact that without the resource, `a` cannot execute its plan, so `a`'s computation would fail.

Agent `b` considers its integrity constraint (5) and has two possibilities: either variable X of the atom in the Δ is equal to `b`, or it is different from `b`. In other words, either it supposes to be the addressee of the request or not. In the first case it should reply `refuse`, as it does not have an available `nail`; however this reply would not be consistent with the hypothesis formulated by `a` that the reply would be `accept`. The Δ would contain both answers `accept` and `refuse` from `b` to `a`, and this is inconsistent

with the integrity constraint (4). The only globally consistent possibility is that `b` is not the addressee of the request. Agent `c` will, in its turn, hypothesize to be the addressee: it will reply `accept`, which is consistent with both ICs.

4.3 A meeting room reservation problem

Speculative computation by Satoh et al. [12] is a technique used to carry on with a distributed computation where information exchange is involved, without waiting for such information to be available. To this purpose, it uses default assumptions on the missing information, and it provides an operational model for the consistency of the overall computation with the assumed defaults – once the information becomes available – or for activating alternative branches in case of inconsistency. The authors present a meeting room reservation problem as an example of computation with defaults, in an environment with unreliable communication. The problem is to organize a meeting among three agents: a, b, and c. If less than two agents attend the meeting, the meeting is cancelled. If exactly two agents attend the meeting, we book a small room. If three agents come, we book a big room.

The problem is modelled in [12] by adopting a master-slave system architecture, where $\{a, b, c\}$ are the *slave* agents, and a master agent m is introduced, whose goal is to reserve the meeting room. By default, m assumes that a and b are available, while c is not. In the reservation process m asks all agents about their availability, while it continues reasoning based on its default assumptions. If m does not receive any answer, a small room is reserved. If m receives an answer that contradicts its assumptions before a certain timeout, e.g., the end of the computation required to solve the top-goal, m backtracks and proceeds accordingly to the received replies.

Roughly speaking, the approach proposed in [12] to this problem is to activate several concurrent processes, each representing a positive or negative assumed answer to a certain question that has been made to the other agents. The processes waiting for an answer are suspended, while those that contain an assumption that is contradicted by an already received answer are killed (in a further refinement of the algorithm [13], such processes are not killed but only suspended, in order to allow further revision of a given answer).

We could model the communication underlying the meeting room reservation problem by means of abduction, and the shared Δ, with a common abducible predicate `free`$/2$. The master agent has the following program[5]:

```
plan(small_room,[X,Y]) ←
  free(X,true), free(Y,true), free(Z,false), X ≠ Y.
plan(big_room,[X,Y,Z]) ←
  free(X,true), free(Y,true), free(Z,true), all_different(X,Y,Z).
plan(cancel_meeting,[]) ←
  free(Y,false), free(Z,false), Y ≠ Z.
← free(X,true), free(X,false).
```

[5] We report only the part related to the replies, which is the most interesting, being the request and the reservation of the room quite straightforward. Also, we associate to the variables `X`, `Y`, `Z` the domain `[a,b,c]`.

For each case (i.e., reserve a small or big room or even cancel the meeting), the master abduces a reply, one for each agent involved in the meeting. For instance, if the (expected) default answer is two agents are available, and one busy, then the master agent plans to reserve a small room, on the basis of the abduced replies (e.g., `free(X, true)`, `free(Y, false)` and `free(Z, true)`), and the computation proceeds.

In the program of the master agent, while exploiting abduction, we can constrain variables `X`, `Y` and `Z` to be all different and each assuming one value among a, b or c. Thus, abduced atoms are ground. However, in our framework, we can also abduce hypotheses with unbound variables, lifting the approach of speculative computation to non ground-terms.

As the computation proceeds, it can be the case that a reply comes from the agents (all, or some of them). In our framework, this reply is abduced by each agent itself, and stored in the common Δ. For each abducible `free(X, true)` a choice point is left open, e.g., `X/a` $\vee$ `X` $\neq$ `a`. If the reply provided by some of the agents, `a`, violates the integrity constraint, i.e., the master agent has assumed the availability of `a`, and this is not the case since `a` is busy, then the master agent has to backtrack, and consider a different set of abducibles. Nonetheless, if no answer comes from the agents, then the default is assumed, once and forever. This framework is able to provide the same answer also in case an atom `free(a, true)` is posted in the blackboard by agent `a` before the room reservation process is started by `m`: in this way, we give a declarative counterpart to a multiple-party speculative computation setting, more general than the one considered in [12], and a framework capable of dealing with non-ground terms.

Furthermore, it is worth noticing that a different operational semantics can be given to abduction, so to recover both synchronous or asynchronous communication: when abducing an atom with non-ground terms, the process can suspend thus miming the behavior of read-only variable of concurrent logic languages (and obtain a synchronous communication), or proceed (asynchronous communication).

5 Abduction and set operations: experiments with $CLP(\mathcal{SET})$

As we noticed in Sect. 3, communication is based on operations on *sets*. For this reason, we decided to perform our experiments on $CLP(\mathcal{SET})$ [8], a constraint language based on sets. $CLP(\mathcal{SET})$ is an instance of the general CLP framework [14] which provides finite sets, along with a few set-based operations, as primitive objects of the language.

Each of the agents could use one of the existing abductive proof procedures (e.g., [15,7,16,17]) to perform abductive reasoning, and produce a set of abducibles. The hypotheses proposed by the various agents, δ_i, could be combined as explained in Sect. 3 with $CLP(\mathcal{SET})$, in order to obtain a globally consistent set of hypotheses Δ (Def. 3).

It is worth noticing that abduction itself can be thought of as based on set operations, thus one may implement also the abductive proof procedure in $CLP(\mathcal{SET})$. In fact, in the abductive computation of each agent, the expected result includes the set δ_i of abduced hypotheses, along with bindings of variables. Each hypothesis is inserted in the set δ_i by an abductive step, affirming that the hypothesis belongs to the set δ_i. The space of possible hypotheses is limited by ICs, that forbid some conjunctions of hypotheses

in δ_i. Consider, for example, the IC:

$$\leftarrow L_1, L_2.$$

where both L_1 and L_2 are abducibles. This constraint limits the possible hypotheses: if δ contains the atom L_1, then it cannot contain L_2 and viceversa.

All the operations on the set δ_i can be defined in terms of two basic operations: the abduction step (making an hypothesis, i.e., $L \in \delta_i$) and the check/propagation of the integrity constraints, that can reject some possible hypotheses (state that some hypotheses cannot belong to δ_i, i.e., $L \notin \delta_i$). Both these operations, the set membership operation and its denial, can be considered as *constraints*, meant to *define* the set δ_i, which is the result of the computation.

We made our experiments by defining a simplified abductive proof procedure in $CLP(\mathcal{SET})$. In our implementation, the abduction of an atom, L_1, is given by two steps. Firstly, we impose that the set δ_i contains the atom L_1, with the constraint $L_1 \in \delta_i$. This will result in the unification $\delta_i = \{L_1|\delta_i'\}$, which, in $CLP(\mathcal{SET})$ syntax, means that $\delta_i = \{L_1\} \cup \delta_i'$.

The second step is imposing integrity constraints. Whenever a new atom is abduced, constraints are imposed on the rest of the set δ_i. In our example, when abducing L_1, we impose that the rest of δ_i should not contain the abducible L_2: $L_2 \notin \delta_i'$. The structure of the predicate responsible for abduction can be the following:

```
abduce(Atom,Delta) ← Delta = {Atom | D1},
   collect_ics(Atom,ICs), impose_ics(ICs,D1,Atom).
```

`collect_ics` collects all the variants of ICs that contain an atom unifying with the abduced atom, together with the corresponding substitution θ. Intuitively, when we abduce a atom L_1, we want to falsify at least one atom in each integrity constraint. `impose_ics` tries to find, in each IC/θ, an atom which is false for all the possible instantiations of its (remaining) universally quantified variables.

It is worth noticing that some of the transitions in the operational semantics of abductive proof procedures are automatically performed by constraint propagation in $CLP(\mathcal{SET})$. For example, proof procedures typically try to find a possibly minimal set of hypotheses, thus they try to unify couples of hypotheses in δ (with transitions called *solution reuse* [16] or *factoring* [7]). Given two hypotheses $p(X)$ and $p(Y)$, abductive proofs unify X and Y, but also consider $X \neq Y$ upon backtracking. In $CLP(\mathcal{SET})$, if δ contains a non-ground atom $p(X)$, i.e., $\delta = \{p(X)|\delta'\}$, when abducing a new atom $p(Y)$ (i.e., imposing the constraint $p(Y) \in \delta$) the nondeterministic propagation provides the two alternative solutions $\delta = \{p(X)|\delta'\}$ with $X = Y$ and $\delta = \{p(X), p(Y)|\delta"\}$ with $X \neq Y$.

6 Related work and discussion

Torroni [18] investigates how to coordinate the abductive reasoning of multiple agents, developing an architecture where several coordination patterns can be chosen. A logic-based language (LAILA) is defined for expressing communication and coordination between logic agents, each one equipped with abductive reasoning capability [5]. LAILA

can be used to model the social behavior of logic-based agents, enabling them to express at a high level several ways to join and coordinate with one another. Our work approaches agent interaction from another perspective: no explicit coordination operators are needed, and the role of abduction is mainly in giving a semantics to interaction seen as information exchange, and not in the agent's internal reasoning. Differently from [5], in this work agent share predicates which are not necessarily ground, and a form of information exchange results from unification and variable binding. This allows for asynchronous interaction patterns, where in principle no agent needs "starting" the distributed proof of a goal, nor coordinating the reasoning activity of others.

Hindriks et al. [19] propose a logic-based approach to agent communication and negotiation where deduction is used to derive information from a received message, and abduction is used to obtain proposals in reply to requests. In particular, deduction serves to derive information from a received message. Abduction serves to obtain proposals in reply to requests. A semantics based on deduction is proposed for the `ask` and `tell` primitives, similarly to other proposals in the literature, while a semantics based on abduction is proposed for the `req` and `offer` primitives. The semantics that they propose, based on the existence of free variables in the communicative acts, shows some similarities with ours; the main difference is that we do not distinguish among the different kinds of communication primitives, and the semantics of communication is uniformly based on abduction.

Sadri et al. [4] propose a framework for agent negotiation based on dialogue, which we sketched in section 4. The work of Sadri et al. [4] differs from ours in its purpose, which is not to give a semantics to agent interaction, but to give an execution model for the activity of the single agent and - based on it - to study formal properties of agents interacting with each other. In [3], the authors represent Kowalski Sadri agents as abducible theories, and formalize communication acts by means of inter-theory reflection theorems, based on the predicate symbols $tell$ and $told$. Intuitively, each time a $tell(a_1, A)$ atom is derived from a theory represented by an agent a_2, the atom $told(a_2, A)$ is consequently derived in the theory represented by a_1, and therefore the proposition A becomes available to it. The framework is provided with a nice formalization, and is based on two particular predicates ($tell/told$) that allow peer-to-peer communication. In our work, we aim to cater for different interaction patterns, possibly involving more than one peer, and to consider communication acts as bi-directional knowledge sharing activities, where several parties may contribute in shaping new information through the unification mechanism.

Satoh et al. [12] present a master-slave system in a setting where communication is assumed to be unreliable, which we briefly introduced in Section 4. The system is given a formal proof-procedure, that consists of two steps: a process reduction phase, and a fact arrival phase. Differently from our work (more focussed on the declarative semantics), speculative computation is an operational model.

Finally, a comment about the proof procedure for abduction. We chose to make our experiments with $CLP(\mathcal{SET})$, that has the advantage that it provides set unification as a first class operation. But there are several abductive proof procedures that could be used instead for our purpose. Of course, our framework requires abduction of non ground atoms, as variables in abducibles can represent request for information.

Denecker and De Schreye [15] introduce a proof procedure for normal abductive logic programs by extending the SLDNF resolution to the case of abduction. More recent work is represented by the SLDNFA(C) system [20] which extends SLDNFA with constraints.

A recent abductive proof procedure dealing with constraints on finite domains is ACLP [16]. ACLP interleaves consistency checking of abducible assumptions and constraint satisfaction. Finally, $\mathcal{A}$-system [17], followup of ACLP and SLDNFA(C), differs from previous two for the explicit treatment of non-determinism that allows the use of heuristic search with different types of heuristics.

7 Conclusions and Future Work

In this work, we presented a framework that gives a uniform treating of abductive reasoning and communication. Groups of abductive agents communicate by abducing non-ground terms and obtain binding for their variables as the result of an (implicit) agreement with other agents. The result of the interaction is modelled as a set of abductive predicates (Δ), consistent with all the local integrity constraints of the agents. We showed some properties of the framework which make it possible to give a semantic characterization to the information exchanged in the abductive process. We presented various examples of communication patterns that can be emulated, like the the dining philosophers and speculative computation. We gave them semantics in terms of abduction and set-based unification. Since the set Δ is constructed by the union of the local hypotheses, we sketched a prototypical implementation in $CLP(\mathcal{SET})$.

In future work, we plan to implement the framework in a fully distributed environment, possibly by exploiting proof procedures based on constraint satisfaction technology. We also plan to provide an operational semantics for our framework, with the semantics of suspension, possibly drawing inspiration from concurrent logic languages.

References

1. Dix, J., Leite, J.A., Satoh, K., eds.: Computational Logic in Multi-Agent Systems: 3rd International Workshop, CLIMA'02, Copenhagen, Denmark, August 1, 2002, Proceedings. Electronic Notes in Theoretical Computer Science **70(5)** (2002)
2. Kowalski, R.A., Sadri, F.: From logic programming towards multi-agent systems. Annals of Mathematics and Artificial Intelligence **25** (1999) 391–419
3. Dell'Acqua, P., Sadri, F., Toni, F.: Combining introspection and communication with rationality and reactivity in agents. In: Logics in Artificial Intelligence, Proc. of JELIA'98. LNCS, Vol. 1489, Springer-Verlag (1998) 17–32
4. Sadri, F., Toni, F., Torroni, P.: An abductive logic programming architecture for negotiating agents. In: Logics in Artificial Intelligence, Proc. of JELIA'02. LNCS, Vol. 2424, Springer-Verlag (2002) 419–431
5. 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
6. 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

7. Fung, T.H., Kowalski, R.A.: The IFF proof procedure for abductive logic programming. Journal of Logic Programming **33** (1997) 151–165
8. Dovier, A., Piazza, C., Pontelli, E., Rossi, G.: Sets and constraint logic programming. ACM Transactions on Programming Languages and Systems **22** (2000) 861–931
9. Dovier, A., Pontelli, E., Rossi, G.: Constructive negation and constraint logic programming with sets. New Generation Computing **19** (2001)
10. Kakas, A.C., Kowalski, R.A., Toni, F.: The role of abduction in logic programming. In Gabbay, D.M., Hogger, C.J., Robinson, J.A., eds.: Handbook of Logic in Artificial Intelligence and Logic Programming. Volume 5., Oxford University Press (1998) 235–324
11. Dijkstra, E.: Hierarchical ordering of sequential processes. Acta Informatica **1** (1971) 115–138
12. 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
13. Satoh, K., Yamamoto, K.: Speculative computation with multi-agent belief revision. In Castelfranchi, C., Lewis Johnson, W., eds.: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002), Part II, Bologna, Italy, ACM Press (2002) 897–904
14. Jaffar, J., Maher, M., Marriott, K., Stuckey, P.: The semantics of constraint logic programs. Journal of Logic Programming **37(1-3)** (1998) 1–46
15. Denecker, M., Schreye, D.D.: SLDNFA: An abductive procedure for normal abductive programs. In Apt, K., ed.: Proceedings of the Joint International Conference and Symposium on Logic Programming, Cambridge, MIT Press (1992) 686–702
16. Kakas, A.C., Michael, A., Mourlas, C.: ACLP: Abductive Constraint Logic Programming. Journal of Logic Programming **44** (2000) 129–177
17. Kakas, A.C., van Nuffelen, B., Denecker, M.: A-System: Problem solving through abduction. In Nebel, B., ed.: Proceedings of the 17th International Joint Conference on Artificial Intelligence, Seattle, Washington, USA, Morgan Kaufmann Publishers (2001) 591–596
18. Torroni, P.: Reasoning and interaction in logic-based multi-agent systems. PhD thesis, Department of Electronics, Computer Science, and Systems, University of Bologna, Italy (2001)
19. Hindriks, K., de Boer, F., van der Hoek, W., Meyer, J.J.: Semantics of communicating agents based on deduction and abduction. In: Foundations And Applications Of Collective Agent Based Systems (CABS). (1999)
20. van Nuffelen, B., Denecker, M.: Problem solving in ID-logic with aggregates. In: Proceedings of the 8th International Workshop on Non-Monotonic Reasoning, NMR'00, Breckenridge, CO. (2000) 1–9

Appendix: Proofs

Proof of Theorem 1. Let us consider the immediate consequence operator, T; we will show that, $\forall n, T^n_{KB_1 \cup KB_2 \cup \Delta} = T^n_{KB_1 \cup \Delta} \cup T^n_{KB_2 \cup \Delta}$.

$T^1_{KB_1 \cup KB_2 \cup \Delta} = T^1_{KB_1 \cup \Delta} \cup T^1_{KB_2 \cup \Delta}$, as it only contains the ground facts in the Δ and in the two KBs.

By induction, let us suppose that $T^n_{KB_1 \cup KB_2 \cup \Delta} = T^n_{KB_1 \cup \Delta} \cup T^n_{KB_2 \cup \Delta}$. By definition of T, since Δ only contains facts,

$$T^{n+1}_{KB_1 \cup \Delta} = T^n_{KB_1 \cup \Delta} \cup \left\{ X : X \leftarrow B \in KB_1, B \subseteq T^n_{KB_1 \cup \Delta} \right\}$$

and analogously for agent A_2.

$$
\begin{array}{l}
T^{n+1}_{KB_1 \cup KB_2 \cup \Delta} = T^{n}_{KB_1 \cup KB_2 \cup \Delta} \cup \\
\quad \cup \left\{ X : X \leftarrow B \in KB_1, B \subseteq T^{n}_{KB_1 \cup KB_2 \cup \Delta} \right\} \cup \\
\quad \cup \left\{ X : X \leftarrow B \in KB_2, B \subseteq T^{n}_{KB_1 \cup KB_2 \cup \Delta} \right\} = \\
= T^{n}_{KB_1 \cup \Delta} \cup \left\{ X : X \leftarrow B \in KB_1, B \subseteq T^{n}_{KB_1 \cup \Delta} \cup T^{n}_{KB_2 \cup \Delta} \right\} \cup \\
T^{n}_{KB_2 \cup \Delta} \cup \left\{ X : X \leftarrow B \in KB_2, B \subseteq T^{n}_{KB_1 \cup \Delta} \cup T^{n}_{KB_2 \cup \Delta} \right\}
\end{array}
$$

Now we only have to show that

$$
\begin{array}{c}
\left\{ X : X \leftarrow B \in KB_1, B \subseteq T^{n}_{KB_1 \cup \Delta} \right\} = \\
= \left\{ X : X \leftarrow B \in KB_1, B \subseteq T^{n}_{KB_1 \cup \Delta} \cup T^{n}_{KB_2 \cup \Delta} \right\}
\end{array}
\tag{6}
$$

(and the same for KB_2).

Eq. 6 is true if for each clause $(X \leftarrow B) \in KB_1$ and each atom $S \in B$, $S \notin T^{n}_{KB_2 \cup \Delta} \setminus T^{n}_{KB_1 \cup \Delta}$. If S belonged to that set, then there would be a clause $(S' \leftarrow B') \in KB_2$ that unifies with S (by definition of $T_{KB_2 \cup \Delta}$), and this is impossible by hypothesis. $\square$

Proof of Theorem 2.

The set Δ is existentially quantified; let us take a ground version of it.

Suppose that the first condition holds. This means that for each abductive logic program i, for each integrity constraint $ic^i_j \in IC_i$ there is an atom a that is not entailed by KB_i:

$$\forall i \, \forall ic^i_j \in IC_i \, \exists a \in ic^i_j : KB_i \cup \Delta \not\models a.$$

If a is abducible, it can be true only if $a \in \Delta$, but this is not the case, since we know that $KB_i \cup \Delta \not\models a$. Since $a \in ic^i_j \in IC_i$, a cannot be defined in any KB_m with $m \neq i$. Thus, we have that a is not entailed by any of the KBs (union Δ):

$$\forall i \, \forall ic^i_j \in IC_i \, \exists a \in ic^i_j : \forall_m KB_m \cup \Delta \not\models a.$$

By Theorem 1, $\forall i \forall ic^i_j \in IC_i \, \exists a \in ic^i_j : \cup_m KB_m \cup \Delta \not\models a$, thus $\forall i \forall ic^i_j \cup_m KB_m \cup \Delta \models ic^i_j$. Since this holds for every integrity constraint, we have that $\cup_m KB_m \cup \Delta \models \cup_i \cup_j ic^i_j$ that is

$$\cup_m KB_m \cup \Delta \models \cup_i IC_i.$$

Viceversa, suppose that the second condition holds. This means that for each agent i, for each integrity constraint $ic^i_j \in IC_i$ there is an atom that is not entailed by the union of the KBs:

$$\forall i \, \forall ic^i_j \, \exists a \in ic^i_j : \cup_m KB_m \cup \Delta \not\models a.$$

By Theorem 1, this is equivalent to

$$\forall i \, \forall ic^i_j \, \exists a \in ic^i_j : \forall_m KB_m \cup \Delta \not\models a$$

In particular, if none of the KBs (union Δ) entails a, even more so neither the KB of the agent i (union Δ) to which ic^i_j belongs entails a:

$$\forall i \, \forall ic^i_j \, \exists a \in ic^i_j : KB_i \cup \Delta \not\models a$$

which means that

$$\forall i \, \forall ic^i_j : KB_i \cup \Delta \models ic^i_j$$

since every integrity constraint of the agent is entailed, also their union is entailed $\square$