# Expressing Collaborative and Competitive Coordination among Abductive Logic Agents

Anna Ciampolini, Evelina Lamma, Paola Mello and Paolo Torroni
University of Bologna
V.le Risorgimento 2, 40136 Bologna, Italy
{aciampolini, elamma, pmello, ptorroni}@deis.unibo.it

## Abstract

This paper presents a language for coordinating several logic-based agents capable of abductive reasoning. The system is particularly suited for solving problems with incomplete knowledge, where agents may need to make reasonable hypotheses about the domain. We defined a simple declarative language to express agent behavior, and in particular, two forms of coordination: collaboration and competition. An example in the area of medical diagnosis is presented to show the features of the language and the behavior of the proposed agent architecture.

## 1 Introduction

Intelligent agents are currently widely used to represent entities with the ability to solve problems, reflecting the results on an environment which might be not completely under their control. Intelligent agents need deductive and pattern-matching capabilities to perform goals and activity requests on them. Recently, a number of works proposed systems where intelligent agents are modeled with logics [Rochefort & al. 99, Kowalski & Sadri 99, Dell'Acqua & Pereira 99, Dell'Acqua & al. 98]. In knowledge-intensive (distributed) applications, it is often the case that agents require some sort of *guess* about a computation which cannot be performed locally since their local knowledge is incomplete or they have to perform some form of hypothetical reasoning. This is very frequent, for instance, in diagnosis systems where each agent has to guess the causes of some symptoms. In this respect, some form of *open* or *abductive* reasoning has to be considered. Abduction has been widely recognized as a powerful mechanism for hypothetical reasoning in presence of incomplete knowledge [Cox & Pietrzykowski 86, Eshgi & Kowalski 89, Kakas & Mancarella 90a], and also captures other important issues such as reasoning with defaults and beliefs [Kowalski 90, Poole 88].

Intelligent multi-agent systems are often characterized by *autonomy* and *social behavior* [Jennings & Wooldridge 98]. Autonomy allows each agent to reason and act independently of the external environment (i.e., of either the user or other agents) making use of its own resources. Nonetheless, this plurality of behaviors needs to be somehow managed, if we want agents to behave effectively as a whole, still autonomously. Thus, the support for agents interaction and coordination should be provided in any multi-agent system. Especially when sophisticated reasoning techniques are needed, it gets useful to keep a separation between internal reasoning infrastructure and their social attitudes. For instance, when each agent locally uses abduction to formulate hypotheses for explaining some observed symptoms, a coordination mechanism for ensuring the consistency of explanations with the knowledge of some other agent might be needed. In order to keep agent's structure as independent as possible of the external environment, a clean separation between intra-agent abductive reasoning and inter-agent coordination should be provided.

In this work, we present a language for coordinating a system of multiple logic abductive agents. The language is supported by *ALIAS* [Ciampolini & al. 99a], a multi-agent architecture where agent reasoning is based on abduction. Our system uses logics for both modeling agent reasoning (in particular abduction, as in [Sadri & Toni 99]), and expressing communication and coordination between agents (as in [Dell'Acqua & al. 99]). In this setting, several autonomous agents, each enclosing a local knowledge base, can either autonomously reason using their own local knowledge base, or can ask other agents to cooperate, in a *collaborative* or *competitive* way, in order to solve a given goal. When several agents are involved in a collaborative computation, each agent, in parallel, must locally solve a given goal using its own knowledge; however, the (abductive) explanations separately found by each agent have to be merged into a unique answer and therefore they have to be consistent with each other. In case of a competitive computation, the same task is assigned to more than one agent. Each agent can provide one (or more) solution for the given task; one solution, among those found by the involved competitive agents, is selected.

The language presented in this paper allows, similarly to other proposals ([Labrou & Finin 94]), to express communication and coordination (with particular regard to collaboration and competition) among agents, using, unlike them, a declarative style.

The paper is organized as follows. Section 2 proposes some preliminary examples in the area of medical diagnosis, with the purpose of showing some typical situations where coordination among intelligent autonomous agents capable of abductive reasoning is required, focusing in particular on collaboration and competition. Section 3 introduces the multi-agent architecture and Section 4 presents the Distributed Abduction Algorithm [Ciampolini & al. 99a] used to perform and coordinate abductive reasoning in a multi-agent context. Section 5 presents *LAILA*, a logic-based language for expressing agent behavior and coordination. Some simple examples are then provided with the purpose of showing how the features of LAILA are mapped on the mechanisms provided by the ALIAS agent architecture. Conclusions and future work follow.

## 2 Some Preliminary Examples

The main objective of our multi-agent architecture (ALIAS) is to support distributed abductive reasoning by joining and coordinating several autonomous intelligent agents, each enclosing a private local knowledge base. In order to let the reader understand the motivations that led us to the development of such architecture and to the definition of the LAILA language, in the following we will discuss some examples taken from the medical diagnosis field, where both patients and specialists can be modeled by means of agents.

Let us consider, for instance, the case of a patient that, in presence of certain symptoms, asks one (or more) doctor(s) for a diagnosis. The doctors make use of abductive reasoning with the purpose of finding diagnostic explanations of the symptoms. In general, we could assume that each agent has a different and partial view of the problem domain. Explanations of symptoms (*i.e.*, diagnoses), are represented by sets of hypotheses, raised by agents as a result of their abductive reasoning.

In such a scenario, it could happen that more than one symptom contemporaneously occur in the patient. Let $S$ be the set of observed symptoms. In case of symptoms that are quite independent of each other, if each symptom could be explained by a different specialist, the patient might ask for a diagnosis of each symptom to a different agent. Each specialist agent, in parallel with others, tries to explain the given symptom using its own knowledge, possibly raising new hypotheses, and on the end it possibly finds several alternative diagnoses for the same symptom. However, the patient aims at obtaining a unique and coherent explanation of the given set of symptoms $S$. To this purpose, each agent can impose integrity constraints about hypotheses: in order to have a *consistent* solution, all of them must be respected. In that case, although each doctor reasons locally in order to find a diagnosis for a single symptom, it is necessary to coordinate all the involved agents when synthesizing the distinct answers produced autonomously by each specialist, into a unique consistent explanation. In the following, we will refer to this form of coordination among agents as to *collaboration*.

A different situation could happen, instead, when a patient wants to contact different doctors in parallel on a given symptom or set of symptoms $s$, with the purpose of collecting different opinions (i.e., diagnoses) about it. The involved agents, handling distinct knowledge bases (possibly contrasting each other) might provide, by means of local reasoning, several alternative diagnoses for $s$ (possibly inconsistent each other). The patient, then, selects only one of them among those available, on the basis of an appropriate policy (e.g. the patient could trust one specialist more than another, or - in case of urgency - he could choose the quickest answer). It is however mandatory that the selected diagnosis is consistent with the patient's knowledge. In situations like this, a different, less strict, form of coordination among agents is required: we will refer to this form of agent interaction as to *competition*.

The previous examples suggest some requirements for an intelligent agent architecture which claims to be suitable for distributed diagnosis purposes. First, each agent should be a separate entity handling a possibly incomplete knowledge: therefore an important feature is the ability of autonomously performing hypothetical reasoning. For this reason, we will describe an agent architecture where agent reasoning is mainly based on abduction. Moreover, agents must be able to coordinate themselves in solving collaborative / competitive goals. Thus, in the proposed architecture autonomous agents could interact following different coordination and communication patterns, including collaboration and competition. Finally, in order to let the user represent and specialize agent behavior, a language for expressing agent knowledge, communication and coordination should be provided. To this purpose, we will define a logic-based language for coordinating agents activities.

## 3 The Architecture of *ALIAS*

The agent architecture we refer to is *ALIAS* (Abductive LogIc Agent System), a system where several intelligent agents, each enclosing a local knowledge base, can either *autonomously* reason on their own local knowledge bases or exhibit a *social* behavior, interacting with other agents following different coordination patterns. A peculiar feature of ALIAS agents is that they can solve a problem by means of abductive reasoning.

The inner structure of each ALIAS abductive agent, shown in Figure 1, is basically composed of two modules: the *Abductive Reasoning Module* ($\mathcal{ARM}$, for short) and the *Agent Behavior Module* ($\mathcal{ABM}$). Both modules encapsulate a local knowledge base (KB): the *abductive* knowledge base (in the $\mathcal{ARM}$ module) and the *behavior* knowledge base (in the $\mathcal{ABM}$ module). The abductive KB is represented by an abductive logic program (for more details, see Section 4); the behavior KB is a set of logic clauses which describe the behavior of the agent within the environment in a declarative style by means of the LAILA language (presented in Section 5). In particular, the social behavior of each agent can be expressed within $\mathcal{ABM}$, by means of explicit communication and collaborative/competitive queries. Each time the agent's behavior requires the abductive explanation of a goal $G$, an interaction between $\mathcal{ABM}$ and $\mathcal{ARM}$ is needed, in order to locally start the abductive proof for $G$. It is up to the $\mathcal{ABM}$ to coordinate the computations carried on by different collaborating/competing agents.

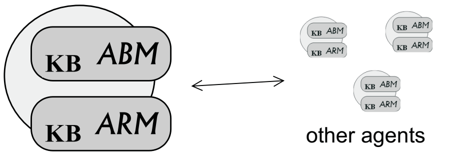


Figure 1: The structure of a ALIAS agent

Within this framework, a multi-agent application is mapped onto several agents, possibly interacting, either collaborating or competing. Being the behavior of each agent modeled via a logic-based language, the computation is driven by goals to be demonstrated.

# 4 Abduction in a Multi-Agent Environment

Abduction is a well known hypothetical reasoning technique [Eshgi & Kowalski 89, Kakas & al. 93, Flach & Kakas 00], that allows to find explanations for a given observation, under the *open world* assumption. In fact, hypothetical reasoning could be extremely useful when the knowledge about the problem domain is incomplete: this is the case of multi-agent applications where each agent might have a partial, and possibly incomplete, view of the world.

In the following we recall some preliminary concepts on abductive reasoning, then we introduce the abduction algorithm adopted by ALIAS agents.

## 4.1 Abductive Logic Programs

As described in 3, each ALIAS agent may enclose (in its $\mathcal{ARM}$ module) an *abductive logic program*.

An abductive logic program is a triple $\langle P, \mathcal{A}, IC \rangle$ where $P$ is a logic program possibly with abducible atoms in clause bodies; $\mathcal{A}$ is a set of *abducible predicates*, i.e., *open* predicates which can be used to form explaining sentences; $IC$ is a set of integrity constraints: each constraint is a denial containing at least one abducible. Given an abductive program $\langle P, \mathcal{A}, IC \rangle$ and a formula $G$, the goal of abduction is to find a (possibly minimal) set of atoms $\Delta$ (i.e., the abductive explanation of $G$) which together with $P$ entails $G$. It is also required that the program $P \cup \Delta$ is consistent with respect to $IC$. According to [Eshgi & Kowalski 89], negation as default, possibly occurring in clause bodies, can be recovered into abduction by replacing negated literals of the form *not a* with a new positive, abducible atom $not\_a$ and by adding the integrity constraint $\leftarrow a, not\_a$ to $IC$. The natural syntactic correspondence between a standard atom and its negation by default is given by the following notion of complement:

$$\bar{l} = \begin{cases} \alpha & \text{if } l = not\_\alpha \\ not\_\alpha & \text{otherwise} \end{cases}$$

where $\alpha$ is an atom.

We suppose that each integrity constraint in $IC$ has at least one abducible in the body. We suppose that abducible predicates have no definition as in [Kakas & Mancarella 90b].

## 4.2 Extending abductive reasoning to multi-agent systems

In a multi-agent setting, we can equip each abductive agent with a distinct abductive logic program. In particular, in ALIAS each agent encloses in its $\mathcal{ARM}$ a local abductive program. Agents can dynamically group into bunches, with the purpose, for instance, of finding the solution of a given goal in a *collaborative* way. In this perspective, although the set of program clauses and integrity constraints might differ from agent to agent, we assume that the set of abducible predicates (default predicates included) is the same for all the agents in a bunch. This implies that when proving a given goal, if an agent $A$ assumes a new hypothesis $h$, all the arguing agents (i.e., the agents belonging to the same bunch) must check the consistency of $h$ with their own integrity constraints. These checks could possibly raise new hypotheses, whose consistency within the bunch has to be recursively checked. Therefore, in ALIAS, the abductive explanation of a goal within a bunch of agents is a set of abduced hypotheses, agreed by all agents in the bunch.

In order to perform abduction in a multi-agent environment we need to introduce some mechanism to support agent bunches, local abduction and global consistency checks. The algorithm we have adopted in the current implementation is based on the (*DAA*, Distributed Abduction Algorithm), discussed in [Ciampolini & al. 99a], which extends towards distribution a proof procedure, defined originally in [Eshgi & Kowalski 89] and further refined in [Kakas & Mancarella 90b], which is correct with respect to the abductive semantics defined in [Brogi & al. 97]. The proof procedure presented in [Kakas & Mancarella 90b] extends the basic resolution mechanism adopted in logic programming by introducing the notion of *abductive* and *consistency* derivation. Intuitively, an *abductive* derivation is the usual logic programming derivation suitably extended in order to consider abducibles. When an abducible atom $h$ is encountered during this derivation, it is assumed, provided this is consistent. The consistency check of a hypothesis, then, starts the second kind of derivation. The *consistency* derivation verifies that, when the hypothesis $h$ is assumed and added to the current set of hypotheses, any integrity constraint containing $h$ fails (i.e., the bodies of all the integrity constraints containing $h$ are false). During this latter procedure, when an abducible $L$ is encountered, in order to prove its failure, an abductive derivation for its complement, $\overline{L}$, is attempted.

The *DAA* algorithm extends the Kakas-Mancarella approach in the sense of distribution: now knowledge is distributed among several agents. In particular, while abductive derivation is limited to the local knowledge base, consistency derivations have to be coordinated within the pool of logic agents of the current bunch. In particular, we adopt a two-phase algorithm, which first collects the results of local[1] abduction processes running the Kakas-Mancarella proof procedure, and then, depending on the chosen agent coordination policy, checks their consistency, in order to find a $\delta$ which is consistent with all the constraints present in the knowledge bases of the agents of the bunch. Thus, the first step involves a local abductive derivation, while the second one involves a consistency derivation in a group of agents. An operational semantics to the notions of abductive and consistency derivation in a multi-agent context is given in 5.2.

It is worth to notice, however, that the ALIAS architecture is not strictly related to the Kakas-Mancarella abductive proof procedure. The same high-level features of the system could exploit different abduction algorithms. In particular, the major drawback of the current approach is that it applies only to ground predicates, thus limiting the real exploitation of the system. Therefore, as a future work, in order to test the system in real applications, we plan to experiment other abductive proof procedures, such as, for instance, *SLDNFA* [Denecker & De Schreye 98], IFF [Fung & Kowalski 97], or the one adopted in the ACLP framework [Kakas 00], a constraint based extension of the Kakas-Mancarella, which makes use of the $ECL^iPS^e$ framework to allow the treatment of non-ground predicates. Moreover, the system could be extended to other forms of inference, such as, for instance, inductive reasoning. To this purpose, we plan also to integrate into the ALIAS architecture agents capable to learn, following an inductive approach.

[1]in this context, *local* means *performed within the agent that is proving the query*, without the intervention of any other agent

# 5 The Coordination Language

In ALIAS, agent behavior is expressed in the *Language for AbductIve Logic Agents* (*LAILA*, for short). This language allows to model agent actions and interactions in a logic programming style. In particular we will focus on agent social behavior, and especially on how each agent can request and coordinate proofs of goals to other agents in the system. To this purpose, in addition to the classical *and* goal composition operator (,), we introduce a *communication* operator (`>`) and two composition operators: the *collaborative* operator (`&`) and the *competitive* operator (`;`) that can be used by each agent to formulate and coordinate queries to other agents. The language provides also a *down-reflection* operator ($\downarrow$), whose role is to trigger a local abductive computation.

In the following subsections we will describe the syntax and the operational semantics of LAILA.

## 5.1 Syntax of LAILA

The syntax of LAILA is given as a BNF grammatics. For the sake of simplicity we provide the sintax for LAILA in the case of propositional programs. Let us consider a system composed by $n+1$ agents. Each agent encapsulates a LAILA program describing its behavior. Let $\mathcal{V}$ be the vocabulary of the language:

$\mathcal{V} = \{\leftarrow,$ `,`, `&` , `;`, `>`, $\downarrow$, $A_0, \ldots, A_n, a_0, \ldots, a_k,$ **not**, **true**, `.` $\}$

where:

- $\leftarrow$ is an implication operator;
- , is the *and* goal composition operator;

- & is the *collaborative* coordination operator;
- ; is the *competitive* coordination operator;
- > is the *communication* operator;
- $\downarrow$ is the *down-reflection* operator;
- $A_i$, $i = 0, \ldots n$ is the identifier of the $(i+1)$-th agent in the system;
- $a_j$, $j = 0, \ldots k$ is an atom (either abducible or not) occurring in the program.

A LAILA program is a set of L-clauses. An L-clause is defined as follows:

$$
\begin{array}{lll}
L-clause & ::= & Atom \ \leftarrow \ Body. \\
Body & ::= & Formula \ ; \ Body \mid \\
 & & Formula \\
Formula & ::= & SFormula \ \& \ Formula \mid \\
 & & SFormula \\
SFormula & ::= & \downarrow ( \ Goal \ ) \mid Agent > ( \ Goal \ ) \mid \\
 & & Goal \\
Goal & ::= & Literal \mid Literal, \ Goal \mid \\
 & & \mathbf{true} \\
Literal & ::= & Atom \mid \mathbf{not} \ Atom \\
Atom & ::= & \mathbf{a_1} \mid \mathbf{a_2} \mid \ \ldots \ \mid \mathbf{a_k} \\
Agent & ::= & \mathbf{A_0} \mid \mathbf{A_1} \mid \ \ldots \ \mid \mathbf{A_n}
\end{array}
$$

A computation can be started by a query, defined as follows:

$$Query \ ::= \ ? \ Body$$

In order to help the reader understand the sense of L-clauses, we anticipate here two simple examples. Let us consider, for instance, the following LAILA competitive query, formulated by agent $A_0$:

$$? \ \downarrow \ \texttt{(G1)} \ ; \ \texttt{A1} \ > \ \texttt{(G2)}$$

It means that $A_0$ must either perform a local abductive derivation for the goal $(G_1)$, *or* ask agent $A_1$ to demonstrate goal $G_2$. Le us consider, now, the following collaborative query, given by agent $A_0$:

$$? \ \texttt{A1} \ > \ \texttt{(G3)} \ \& \ \texttt{A2} \ > \ \texttt{(G4)}$$

It means that agent $A_0$ asks agent $A_1$ to prove $G_3$ and $A_2$ to prove goal $G_4$.

## 5.2 Operational Semantics of LAILA

In this section we present LAILA operational semantics. For the sake of simplicity, the following inference rules refer to the case of propositional LAILA programs. A LAILA program $P$ is a collection of L-clauses, possibly distributed among a set of agents. In the following, $A_0, \ldots, A_n$ denote agents in the system; $f$ denotes a single formula, $F$ a composition of formulae; $\delta_1, \ldots \delta n$ denote conjunctions of abduced hypotheses; $G$ represents a goal and $g$ denotes an atomic goal; $L$ is a literal. Given a formula $F$, let $b(F)$ be the set of agents involved in message exchanges by $F$:

$b(F) = \{A \mid A$ is an agent in a communication formula $\in F\}$
For instance, given the formula $F$ of $A_0$'s $\mathcal{ABM}$:

$$A_1 \ > \ (G_1); \ A_2 \ > \ (G_2) \ \& \ A_3 \ > \ (G_3),$$

$b(F)$ is $\{A_0, \ A_1, \ A_2, \ A_3\}$.

Let us introduce the definition of a successful top-down derivation.

**Definition 1 (Successful top-down derivation)** *Let $P$ be a program and $F$ a formula. a* top-down derivation *for $F$ in $P$ can be traced in terms of (possibly infinite) sequences of steps: $A \vdash_{\delta_{in,i},\delta_{out,i}} F_i$, where $A$ is an agent, $\delta_{in,i}$ and $\delta_{out,i}$ are sets of abduced hypotheses, and $F_i$ is a formula. Each step is obtained by applying within $A$ a suitable inference rule starting from the set $\delta_{in,i}$ of hypotheses, and possibly producing a new set of hypotheses $\delta_{out,i}$. The first step of a top-down derivation starts from the empty set $\delta_{in,0}$. A* top-down *derivation is successful if, at some step $k$, the null formula is derived. The set $\delta_{out,k}$ represents the obtained abductive explanation associated with the successful derivation.*

In the following:

- $A_i \overset{abd}{\vdash}_{\delta_i \delta_j} s$, where $s$ is a set of atoms, denotes the local abductive proof of the conjunction of all atoms in $s$, perfomed by agent $A_i$ (whose meaning is given by the adopted abductive proof procedure);
- $B \overset{cons}{\vdash}_{\delta_i \delta_j} s$, where $s$ is a set of atoms, denotes the consistency check of the conjunction of all atoms in $s$, with respect to the integrity constraints of all agents in bunch $B$.

Let us give the set of inference rules modeling the operational behavior of the system.

**Definition 2 (Down reflection formula)**

$$\frac{A \overset{abd}{\vdash}_{\delta_1,\delta_2} \ G}{A \vdash_{\delta_1,\delta_2} \ \downarrow (G)}$$

Therefore, the goal $\downarrow (G)$ triggers a local abductive derivation for $G$.

**Definition 3 (Communication formula)**

$$\frac{A_1 \vdash_{\delta_1,\delta_2'} G \quad \wedge \quad \{A_0, A_1\} \overset{cons}{\vdash}_{\delta_1,\delta_2} \ \delta_2'}{A_0 \vdash_{\delta_1,\delta_2} \ A_1 > (G)}$$

**Definition 4 (Collaborative formula)**

$$\frac{A_0 \vdash_{\delta_1,\delta_2'} f \wedge A_0 \vdash_{\delta_1,\delta_2''} F \wedge \{A_0\} \cup b(f\&F) \overset{cons}{\vdash}_{\delta_1,\delta_2} \quad \delta_2' \cup \delta_2''}{A_0 \vdash_{\delta_1,\delta_2} f\&F}$$

Thus, the following query, (formulated, for instance, by agent $A_0$):

$$?\ \mathtt{A_1} > (\mathtt{q_1})\ \ \&\ \ \mathtt{A_2} > (\mathtt{q_2})$$

has the following effects:

- $A_0$ asks $A_1$ to solve $q_1$; if $q_1$ succeeds, $N$ ($N > 0$) abductive explanations $\delta_{1,i}$ ($i \in [1, \ldots, N]$) for $q_1$ could be obtained.

- $A_0$ asks $A_2$ to solve $q_2$ ; if $q_2$ succeeds, $M$ ($M > 0$) abductive explanations $\delta_{2,j}$ ($j \in [1, \ldots, M]$) for $q_2$ could be obtained.

The abductive explanation for the query is therefore a set of hypotheses $\delta$ including both $\delta_{1,i}$ and $\delta_{2,j}$ (for some $i \in [1, \ldots, N]$, $j \in [1, \ldots, M]$), such that it is consistent in the bunch $\{A_0, A_1, A_2\}$. If such $\delta$ cannot be obtained, or if either $A_1 > q_1$ or $A_2 > q_2$ fails, the query $Q$ fails.

**Definition 5 (Competitive formula)**

$$\frac{(A_0 \vdash_{\delta_1,\delta_2'} f \quad \vee \quad A_0 \vdash_{\delta_1,\delta_2''} F) \quad \wedge \quad \delta_2 \in \{\delta_2', \delta_2''\}}{A_0 \vdash_{\delta_1,\delta_2} f; F}$$

For instance, let us consider the following *competitive* query, formulated by agent $A_0$:

$$?\ \mathtt{A_1} > (\mathtt{q_1})\quad ;\ \ \mathtt{A_2} > (\mathtt{q_2})$$

it causes the following effects :

- $A_0$ asks $A_1$ to solve $q_1$; if $q_1$ succeeds, $N$ ($N > 0$) abductive explanations $\delta_{1,i}$ ($i \in [1, \ldots N]$) for $q_1$ could be obtained.

- $A_0$ asks $A_2$ to solve $q_2$ in the bunch $\{A_0, A_2\}$; if $q_2$ succeeds, $M$ ($M > 0$) abductive explanations $\delta_{2,j}$ ($j \in [1, \ldots M]$) for $q_2$ could be obtained.

The resulting abductive explanation is either $\delta_{1,i}$ (for some $i \in [1, \ldots, N]$) or $\delta_{2,j}$ (for some $j \in [1, \ldots, M]$). It is worth to notice that in this case the selection rule for the abductive explanation $\delta$ is not specified: it could be either purely non-deterministic or it could follow a particular criterion (*e.g.*, priority among the components of the query). If both $q_1$ and $q_2$ fail, the competitive query fails.

**Definition 6 (Consistency check)**

$$\frac{\forall A_i \in B \quad A_i \overset{abd}{\vdash}_{\delta_1,\delta_2^i} \delta \quad \wedge \quad B \overset{cons}{\vdash}_{\delta_1,\delta_2} \quad \bigcup_i \delta_2^i}{B \overset{cons}{\vdash}_{\delta_1,\delta_2} \quad \delta}$$

$$\frac{}{B \overset{cons}{\vdash}_{\delta_1,\delta_1} \quad \delta}$$

Finally, the semantics of goal formulas is described by the following inference rules:

**Definition 7 (Goal formula)**

$$\frac{A \vdash_{\delta_1,\delta_{2'}} h \quad \wedge \quad A \vdash_{\delta_{2'},\delta_2} G}{A \vdash_{\delta_1,\delta_2} h, G}$$

$$\frac{\exists h \leftarrow G\ \in A \quad \wedge \quad A \vdash_{\delta_1,\delta_2} G}{A \vdash_{\delta_1,\delta_2} h}$$

$$\frac{}{A \vdash_{\delta,\delta} \quad \mathtt{true}}$$

# 6 An Example

```
Abducibles : {diagnose/2, negated atoms}

ABM :

symptom(Patient, vomit, X) ←
    ↓ ( symptom(Patient, vomit, X) ).
symptom(Patient, general_conditions, X) ←
    ↓ ( symptom(Patient, general_conditions, X) ).
    ...

ARM :

symptom(Patient, vomit, relentless) ←
   diagnose(Patient, pancreatitis).
symptom(Patient, vomit, relentless) ←
   diagnose(Patient, gastric_ulcer).
symptom(Patient, general_conditions, very_bad) ←
   diagnose(Patient, pancreatitis).
symptom(Patient, general_conditions, very_bad) ←
   diagnose(Patient, gastric_ulcer).
    ...
young(peter).
    ...
← diagnose(Patient, pancreatitis, young(Patient)).
    ...
```

Figure 2: The gastroenterology agent knowledge base

Let us consider an hospital where patients go to the reception before they are led to wards. Both the

$\mathcal{A}bducibles$ : $\{$diagnose/2, *negated atoms*$\}$

$\mathcal{ABM}$ :

```
symptom(Patient, vomit, X) ←
    ↓ ( symptom(Patient, vomit, X) ).
    ...
```

$\mathcal{ARM}$ :

```
symptom(Patient, vomit, _) ←
   diagnose(Patient, flu).
symptom(Patient, vomit, _) ←
   diagnose(Patient, pregnancy).
    ...
female(mary).
    ...
← diagnose(Patient, pregnancy),
   not female(Patient).
← diagnose(Patient, flu),
   symptom(Patient, general_conditions, very_bad).
← diagnose(Patient, flu),
   symptom(Patient, itch, sine materia).
    ...
```

Figure 3: The general medicine agent knowledge base

hospital reception and the wards are provided with agents capable of abductive reasoning, whose aim is to accept the patient and to address him / her to a relevant ward. The patient is thus asked the symptoms; general data like age, sex, past diseases etc. are considered facts that the hospital might contain in its database or might add on demand. The agents reason on symptoms and generate hypotheses of diagnosis, which can be further used by the hospital structures.

In this example we consider three agents, each specialized in a particular area (gastroenterology, general medicine, dermatology) plus one at the reception (receptionist agent), whose task is to translate the patient symptoms into LAILA queries. In this framework, each specialist agent contains in its ABM a set of LAILA clauses, whose heads show the symptoms that the specific agent is able to interpret. We assume that the receptionist agent uses this knowledge to formulate an appropriate query for the patient. In Figures 2, 3, 4 we show the specialists' knowledge bases.

As an example of the competitive case, let us consider a patient, Peter, who complains of having relentless vomit and shows to have very bad general conditions. Peter is young and male. The reception agent generates a query for the gastroenterology agent and the general medicine agent; we suppose that the generated query is competitive:

$\mathcal{A}bducibles$ : $\{$diagnose/2, *negated atoms*$\}$

$\mathcal{ABM}$ :

```
symptom(Patient, itch, X) ←
    ↓ ( symptom(Patient, itch, X) ).
    ...
```

$\mathcal{ARM}$ :

```
symptom(Patient, itch, sine materia) ←
   diagnose(Patient, pregnancy).
    ...
female(mary).
    ...
← diagnose(Patient, pregnancy),
   not female(Patient).
← diagnose(Patient, pregnancy),
   symptom(Patient, itch, sine materia),
   pregnant(Patient, initial months).
    ...
```

Figure 4: The dermatology agent knowledge base

*Peter's competitive query*

```
?gastroenterology  >
   (symptom(peter, vomit, relentless),
   symptom(peter, general_conditions, very_bad));
  general_medicine  >
   (symptom(peter, vomit, relentless)).
```

The gastroenterology agent starts an abductive derivation aimed at demonstrating the two symptoms, and eventually diagnoses the illness as gastric ulcer:

$$\delta_{\texttt{Gastr}} = \{\texttt{diagnose(peter, gastric\_ulcer)}\}$$

At the same time, the general medicine agent produces an abductive explanation to the first symptom, resulting in another set of hypotheses:

$$\delta_{\texttt{GenMed}} = \{\texttt{diagnose(peter, flu)}, \texttt{not symptom(peter, general\_conditions, very\_bad)}\}$$

The receptionist agent accepts $\delta_{Gastr}$ as a plausible explanation for his symptoms, and addresses Peter to the gastroenterology ward, while it discards $\delta_{GenMed}$ because it is inconsistent with Peter's symptoms, since it assumes that the patient's general conditions can not be very bad, as they actually are (it is in contrast with the query).

As an example of the collaborative case, let us consider another patient, Mary, who complains of having relentless vomit and hitch without evidence on the skin (it is called *sine materia*). Mary is female. The reception agent generates a query for the general medicine agent and for the dermatology agent; we suppose that the generated query is collaborative because none of the two agents is able to deal with all of Mary's symptoms. The query is as follows:

*Mary's collaborative query*

```
?general_medicine >
  ( symptom(mary, vomit, relentless) ) &
  dermatology >
  ( symptom(mary, hitch, sine_materia) ).
```

The general medicine agent starts an abductive derivation aimed at demonstrating the first symptom, and eventually diagnoses flu:

$$\delta_{\mathtt{Gastr}} = \{\mathtt{diagnose(mary, flu)}, \mathtt{not\ symptom(mary, general\_conditions, very\_bad)}, \mathtt{not\ symptom(mary, itch, sine\_materia)}\}$$

At the same time, the dermatology agent produces an abductive explanation to the second symptom, resulting in another set of hypotheses:

$$\delta_{\mathtt{Dermat}} = \{\mathtt{diagnose(mary, pregnancy)}, \mathtt{not\ pregnant(mary, initial\_months)}\}$$

Next, the consistency check of the two deltas reveals that it is not an acceptable answer, because the first delta is in contrast with the query itself. A backtracking mechanism is issued, asking the two agents other possible explanations of the symptoms. Eventually, the general medicine agent returns a new delta which is coherent with the dermatology agent's delta:

$$\delta^1_{\mathtt{GenMed}} = \{\mathtt{diagnose(mary, pregnancy)}\}$$

The receptionist agent accepts the union of the two deltas, $\delta^1_{GenMed} \cup \delta_{Dermat}$ as a plausible explanation for Mary's symptoms.

# 7 Conclusion And Further Work

In this work we presented LAILA a language for expressing communication and coordination among logic-based agents in a declarative style. Agents are thought to interact within a system, ALIAS, whose current implementation allows distributed abduction to be performed among dynamically grouped agents [Ciampolini & al. 99a, Ciampolini & al. 99b].

We are improving the implementation of ALIAS in order to support the LAILA coordination language. Also, we would like to provide a declarative semantics for the language. Moreover, we plan to extend the architecture with the purpose of coping with other abductive proof procedures and other forms of reasoning, and in particular, with regard to inductive reasoning. We would like to investigate the issue of knowledge merging [Meyer 00] and to put it in relation with the semantics given to the collaborative and competitive operators, when combining the results of multiple LAILA queries. Our intention is also to test the system in a real world case, in particular implementing distributed diagnosis applications.

# 8 Acknowledgements

This work has been supported by M.U.R.S.T. Project on *Intelligent agents: interaction and knowledge acquisition.*

# References

[Ciampolini & al. 99a] A. Ciampolini, E. Lamma, P. Mello and P. Torroni. An Implementation for Abductive Logic Agents. In *Proc. AI*IA99*, Springer-Verlag LNAI 1792 (to appear).

[Ciampolini & al. 99b] A. Ciampolini, E. Lamma, P. Mello and P. Torroni. Rambling Abductive Agents in *ALIAS*. In [Rochefort & al. 99].

[Brogi & al. 97] A. Brogi, E. Lamma, P. Mancarella and P. Mello. A Unifying View for Logic Programming with Non-Monotonic Reasoning. In *Theoretical Computer Science*, Vol. 184, 1–49, North Holland, 1997.

[Cox & Pietrzykowski 86] P. T. Cox and T. Pietrzykowski. Causes for events: Their computation and applications. In *Proc. CADE-86*, 608, 1986.

[Dell'Acqua & Pereira 99] P. Dell'Acqua and L. M. Pereira. Updating Agents. In [Rochefort & al. 99].

[Dell'Acqua & al. 98] P. Dell'Acqua, F. Sadri and F. Toni. Combining Introspection and Communication with Rationality and Reactivity in Agents. In U. Furbach and L. Farinas del Cerro eds., *Proc. 6th European Workshop on Logics in Artificial Intelligence*, Springer Verlag LNAI 1489, 17–32 (1998)

[Dell'Acqua & al. 99] P. Dell'Acqua, F. Sadri and F. Toni. Communicating Agents. In [Rochefort & al. 99].

[Denecker & De Schreye 98] M. Denecker and D. De Schreye SLDNFA: an abductive procedure for abductive logic programs. *Journal of Logic Programming*, 34(2):111–167, Elsevier, 1998.

[Flach & Kakas 00] P. A. Flach and A. C. Kakas. *Abduction and Induction*. Applied logic series, 18. Kluwer Academic Publishers, Dodrecht, The Netherlands, 2000.

[Fung & Kowalski 97] T. H. Fung and R. Kowalski. The IFF proof procedure for abductive logic programming. In Journal of Logic Programming, 33(2), pages 151–165, 1997.

[Eshgi & Kowalski 89] K. Eshgi and R. A. Kowalski. Abduction compared with negation by failure. In G. Levi and M. Martelli, editors, *Proc. 6th Int. Conf. on Logic Programming*, 234. MIT Press, 1989.

[Kakas 00] A. C. Kakas. ACLP: integrating abduction and constraint solving. *Proc. NMR'00*, Breckenridge, CO, 2000.

[Kakas & al. 93] A. C. Kakas, R. A. Kowalski and F. Toni. Abductive Logic Programming. *Journal of Logic and Computation*, 2(6):719–770, 1993.

[Kakas & Mancarella 90a] A. C. Kakas and P. Mancarella. Generalized stable models: a semantics for abduction. In *Proc. 9th European Conf. on Artificial Intelligence*. Pitman Pub., 1990.

[Kakas & Mancarella 90b] A. C. Kakas and P. Mancarella. On the relation between Truth Maintenance and Abduction. In *Proc. PRICAI90*, 1990.

[Kowalski 90] R. A. Kowalski. Problems and promises of computational logic. In *Proc. Symp. on Computational Logic*, 1-36. Springer-Verlag, Nov. 1990.

[Kowalski & Sadri 99] R. A. Kowalski and F. Sadri. From Logic Programming to Multi-Agent Systems. In *Annals of Mathematics and Artificial Intelligence*, 1999 (to appear).

[Jennings & Wooldridge 98] N. R. Jennings and M. J. Wooldridge, eds., *Agent Technology*. Springer-Verlag, 1998.

[Labrou & Finin 94] Y. Labrou and T. Finin. A semantics approach to KQML – a general purpose communication language for software agents. In *Proc. 3rd Int. Conf. on Information and Knowledge Management*, 1994.

[Meyer 00] T. Meyer. On the semantics of merging. *Proc. NMR'00*, Breckenridge, CO, 2000.

[Poole 88] D. L. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36:27. Elsevier, 1988.

[Rochefort & al. 99] S. Rochefort, F. Sadri, and F. Toni, eds., *Proc. Int. Workshop on Multi-Agent Systems in Logic Programming*. In conjunction with ICLP'99, Las Cruces, New Mexico, 1999.

[Sadri & Toni 99] F. Sadri and F. Toni. Abduction with Negation as Failure for Active Databases and Agents. In *LNAI, Proc. AI*IA99*. Springer-Verlag, 1999 (to appear).