# Agent Programming using Defeasible Argumentation for Knowledge Representation and Reasoning

**Sebastián Gottifredi  Alejandro J. García  Guillermo R. Simari**
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET),
Laboratorio de Investigación y Desarrollo de Inteligencia Artificial,
Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur,
Avenida Alem 1253,(B8000BCP), Bahía Blanca, Argentina
Tel: (0291) 459-5135 / Fax: (0291) 459-5136
Email: {sg,ajg,grs}@cs.uns.edu.ar

**Abstract**

In this work two declarative approaches based on the BDI theory are studied, an agent programming language 3APL and an agent architecture that uses defeasible argumentation for knowledge representation and reasoning. Based on that study and considering that in 3APL the knowledge representation language is not fixed, we will propose 3APL-DeLP where the agent knowledge is represented by a DeLP-program and the agent may reason with a defeasible argumentation formalism.

Keywords: Agent Programming Languages, Agent Architectures, Agent Theories, Defeasible Argumentation.

## 1 INTRODUCTION

In this work two declarative approaches based on the BDI theory are studied. The 3APL programming language, was chosen because it allows a declarative specification and implementation of the agents, where agents can communicate and can reach their goals using plans. The agent architecture proposed in [7],[6] (BDI-DeLP), was chosen because it uses Defeasible argumentation for the knowledge representation and reasoning of the agents. Based on the study of the approaches and considering that in 3APL the knowledge representation language is not fixed, we will propose 3APL-DeLP where the agent knowledge is represented by a DeLP-program and the agent may reason with a defeasible argumentation formalism.

The importance of using intelligent agents based on mental components like Beliefs, Desires, Commitments and Intentions to solve complex problems is well known in the literature [11], especially those agents based on BDI theory [1]. Nowadays, tools are needed to specify and programs agents in terms of these components. Several programming Languages and architectures based on BDI are proposed in the literature. However, only some of them allow to specify agents in a declarative way. In particular, we are interested in agent development tools that provide an argumentative mechanism for agent reasoning, besides a declarative way to specify its mental components.

This work is organized as follows. In Section 2 an introduction to 3APL and BDI-DeLP approaches is made, indicating goals, advantages and scope for each of them. In Section 3 examples for the approaches are shown. Then an analysis of how the mental components are specified and how reasoning process works for each approach is made. In Section 4 3APL-DeLP is specified. Finally, in Section 5 we expose some conclusions and areas of future research.

Partially supported by CONICET (PIP 5050), Universidad Nacional del Sur and Agencia Nacional de Promoción Científica y Tecnológica.

# 2 TWO APPROACHES BASED ON THE BDI THEORY

In this section, two approaches that use BDI-Theory [1], are introduced. As exposed in [11] the research on agent programming can be divided in three areas: agent theories, agent architectures and agent programming languages. Agent theories are related to the formalisms that specify the properties of the agents. Agent architectures are related to the building of mental components and their interactions, satisfying the properties of an agent theory. Finally, agent programming languages are related to the primitives used to program and implement agents specified by agent theory.

This work is focused on the BDI agent theory. This theory exposes that the mental state of an agent is composed by three components which try to capture the intuitive meaning of Beliefs, Desires and Intentions. The BDI theory is well known in the literature and several agent architectures and Languages have been defined using it. In this work, two of these approaches will be considered: an agent architecture called BDI-DeLP, and an agent programming language called 3APL.

## 2.1 A BDI agent architecture with a Dialectical Framework: BDI-DeLP

In [7] and [6] a BDI agent architecture based on a Dialectical Framework was defined. The goal of this proposal is to allow the specification of the agent mental components and their relations, using Defeasible Logic Programming (DeLP) [4]. This architecture is based on the BDI theory. It introduces a formalism to allow the specification of Beliefs, Desires and Intentions using DeLP programs. Thus, the agent may reason with a defeasible argumentation formalism. Therefore this proposal will be referred as BDI-DeLP in the rest of this work.

BDI-DeLP has advantages as an agent architecture: BDI agents mental components and their interaction can be specified in a declarative way (see Fig.1), and it uses a high level mechanism (DeLP) for knowledge representation and reasoning.

Next, we give a brief summary of DeLP (for more details see [4]). The goal of this summary is to introduce the concepts of DeLP that will be used in the analysis of further sections. In DeLP, knowledge is represented using facts, strict rules, and defeasible rules:

- *Facts* are ground literals representing atomic information or the negation of atomic information using strong negation "$\sim$".
- *Strict Rules* are denoted $L_0 \leftarrow L_1, \ldots, L_n$, where $L_0$ is a ground literal and $\{L_i\}_{i>0}$ is a set of ground extended literals ($e.g.\ \sim a \leftarrow b$).
- *Defeasible Rules* are denoted $L_0 \prec\!\!- L_1, \ldots, L_n$, where $L_0$ is a ground literal and $\{L_i\}_{i>0}$ is a set of ground extended literals. ($e.g.\ c \prec\!\!- \sim d \quad f \prec\!\!- not\ g$).

Rules are distinguished by the type of arrows, and a defeasible rule "$Head \prec\!\!- Body$" expresses that "*reasons to believe in the antecedent* $Body$ *give reasons to believe in the consequent* $Head$" representing tentative information that may be used if nothing could be posed against it.

A Defeasible Logic Program (DeLP) $\mathcal{P}$ is a set of facts, strict rules and defeasible rules. When required, $\mathcal{P}$ is denoted $(\Pi, \Delta)$ distinguishing the subset $\Pi$ of facts and strict rules, and the subset $\Delta$ of defeasible rules. Strict and defeasible rules are ground, however, following the usual convention, some examples will use "schematic rules" with variables.

*Strong negation* could appear in the head of program rules, and can be used to represent contradictory knowledge. From a program $(\Pi, \Delta)$ contradictory literals could be derived. however, the set $\Pi$ (used to represent non-defeasible information) must be non-contradictory, $i.e.$no pair of contradictory literals can be derived from $\Pi$. Given a literal $L$, $\overline{L}$ represents the complement with respect to strong negation.

If contradictory literals are derived from $(\Pi, \Delta)$, a dialectical process is used for deciding which literal prevails. In short, an *argument* for a literal $L$, denoted $\langle \mathcal{A}, L \rangle$, is a minimal set of defeasible

rules $\mathcal{A} \subseteq \Pi$, such that $\mathcal{A} \cup \Pi$ is non-contradictory, and there is a derivation for $L$ from $\mathcal{A} \cup \Pi$. A literal $L$ is *warranted* from $(\Pi, \Delta)$ if there exists a non-defeated argument $\mathcal{A}$ supporting $L$. To establish if $\langle \mathcal{A}, L \rangle$ is a non-defeated argument, *argument rebuttals* or *counter-arguments* that could be *defeaters* for $\langle \mathcal{A}, L \rangle$ are considered, $i.e.$counter-arguments that by some criterion are preferred to $\langle \mathcal{A}, L \rangle$. A defeater $\mathcal{A}_1$ for an argument $\mathcal{A}_2$ can be proper ($\mathcal{A}_1$ *stronger than* $\mathcal{A}_2$) or *blocking* (same strength). In the examples of this paper we assume generalized specificity as the comparison criterion, however, as explained in [4] the criterion could be easily changed.

Since defeaters are arguments, there may exist defeaters for them, and defeaters for these defeaters, and so on. Thus, a sequence of arguments called *argumentation line* is constructed, where each argument defeats its predecessor in the line (for a detailed explanation of this dialectical process see [4]). In DeLP, a query $Q$ could have four possible answers: YES, if $Q$ is warranted; NO, if the complement of $Q$ is warranted; UNDECIDED, if neither $Q$ nor its complement is warranted; and UNKNOWN, if $Q$ is not in the signature of the program.

### 2.2 An agent programming Language: 3APL

In contrast with the BDI-DeLP that provides an agent architecture, 3APL [5] provides an agent programming language. 3APL was presented in [5] and extended in [3] and follows the spirit of Shoham's Agent Oriented Programming paradigm [8]. Its goal is to allow the programmer the creation of cognitive agents that can interact each other. As an agent programming language 3APL [11] is based on the BDI agent theory, and provides a concrete language to program and execute agents. The agents of 3APL combine logic programming (for the specification of their mental components) and imperative programming (for the structure of their plans).

3APL has advantages as an agent programming language: Agents can be specified and implemented in a declarative way through their mental components (see Fig.2), these mental components follow the BDI mental components, agents can solve their goals via plans and communication between agents can be specified and affects the agent mental components. The authors of 3APL have developed a platform[9] were agents can executed and tested.

## 3 AN ANALYSIS OF 3APL AND BDI-DELP

In this section an analysis of 3APL and BDI-DeLP will be made. The goal of this analysis is to expose strong and weak points of each approach, and show how these alternatives can complement each other.

This analysis will be centered in knowledge representation and Reasoning. Therefore, the analysis will be divided in two categories: Mental Components and Reasoning mechanism where the mental component area will be divided in three subcategories: Beliefs, Desires and Intentions.

Since 3APL is agent programming language, it provides support for full agent implementation. Thus, communication and planning, will not be analysed in deep in this work because these items are only supported by 3APL. The communication model used in 3APL is presented in [10] and is based on the *FIPA* standards. This model is integrated with mental components of the agent. Plans in 3APL can be a basic action or a structure of basic actions. This structure follows the spirit of an imperative programming language.

Now we will introduce a working example in order to show and analyze the main differences between both approaches.

**Example 1** *Consider a cognitive agent with the following features:*

- *it has five signal sensors for perceiving the environment. If $s_i$ is present in the agent beliefs means that the sensor $i$ is active.*
- *it can infer new beliefs from active sensors: $b_1$, if $s_1$ or $s_2$ are active; $b_2$ if $s_3$ is active; $\sim b_1$, if $s_1$ and $s_2$ are both active, or if $s_5$ is active; and $\sim b_2$, if $s_2$ and $s_4$ are both active.*
- *it has two desires: $g_1$, achievable if it believes in $b_1$ and not achievable if it believes in $b_2$; and $g_2$, achievable if $s_3$ is active and not achievable if it believe in $b_1$.*
- *it has: two ways to achieve the desire $g_1$, one when $s_1$ and $s_4$ are both active, and the other only $s_4$ is active; and one way to achieve the desire $g_2$, that is when $s_4$ is active.*

The agent of Ex.1 has not necessarily a real connotation, it is a brief example that will be used just to show the representational capabilities of both approaches. Next, we will show a specification for the agent of Ex.1 in BDI-DeLP (Fig.1) and in 3APL (Fig.2). The goal of these figures is to show similarities and differences between both approaches. This specification will represent a snapshot of the agent mental state in an arbitrary moment, where the sensors $s_1$, $s_3$, $s_4$, $s_5$ are active. The mental components of the agent of each figure will be detailed below.

The specification of the agent of Ex.1 with BDI-DeLP is shown below in Fig.1. In this specification mental components like, Beliefs ($\Phi$ and $BR$), Desires ($D$), filtering rules for the desires ($\mathcal{P}_{\mathcal{F}}$) and the intention rules ($IR$), can be differentiated. There, $\Phi$ has facts used to represents sensors, $BR$ has rules used to obtain inferred beliefs, $D$ has facts used to represents the desires, $\mathcal{P}_{\mathcal{F}}$ has rules used to determine if a desire is achievable, and $IR$ has rules used to determine which desire the agent will try to achieve next.

$$\Phi = \left\{ \begin{array}{l} s_1 \\ s_3 \\ s_4 \\ s_5 \end{array} \right\} \quad BR = \left\{ \begin{array}{l} b_1 \prec s_1 \\ b_1 \prec s_2 \\ \sim b_1 \prec s_5 \\ \sim b_1 \prec s_1, s_2 \\ b_2 \prec s_3 \\ \sim b_2 \prec s_2, s_4 \end{array} \right\} \quad D = \left\{ \begin{array}{l} g_1 \\ g_2 \end{array} \right\}$$

$$\mathcal{P}_{\mathcal{F}} = \left\{ \begin{array}{l} g_1 \prec b_1 \\ \sim g_1 \prec b_2 \\ g_2 \prec s_3 \\ \sim g_2 \prec \sim b_1 \end{array} \right\} \quad IR = \left\{ \begin{array}{l} g_1 \Leftarrow \{s_1, s_4\}\{\} \\ g_1 \Leftarrow \{s_4\}\{\} \\ g_2 \Leftarrow \{s_4\}\{\} \end{array} \right\}$$

Figure 1: BDI-DeLP Agent example

The specification of the agent of Ex.1 with 3APL is shown below in Fig.2. In this specification, Beliefs (`BELIEF BASE`), capabilities to manipulate the beliefs (`CAPABILITIES`), Desires (`GOAL BASE`) and the reasoning rules (`RULEBASE`), can be differentiated. There, `BELIEF BASE` has facts and rules used to represent the sensors and the inferred beliefs, `CAPABILITIES` has rules used to represent updates of the sensors, `GOAL BASE` has facts used to represent the desires, and `RULEBASE` has rules used to select a desire and a plan for achieving it.

In Fig.2 in order to obtain an equivalent version of the agent specified in Fig.1, the defeasible rules and the argumentation process were emulated. This will be described in detail in the rest of the section.

```
CAPABILITIES {
    {perception(Y)} Add(X) {not perception(Y), perception(X)}
} BELIEFBASE {
   perception([s1, s3, s4, s5]),
   perceived(X):-perception(L),member(X,L).
   b1:- perceived(s1), not tail1, not perceived(s5).
   b1:- perceived(s2), not tail1, not perceived(s5).
   tail1:- perceived(s1), perceived(s2).
   b2:- perceived(s3), not tail2.
   tail2:- perceived(s2), perceived(s4).
} GOALBASE {
    g1, g2
} RULEBASE {
    g1 <- perceived(s1), perceived(s4), b1, not(b2) | {Plan1Goal1, Sense(X), Add(X)},
    g1 <- perceived(s4), b1, not(b2) | {Plan2Goal1, Sense(X), Add(X)},
    g2 <- perceived(s4), perceived(s3), not(tail1) | {PlanGoal2, Sense(X), Add(X)},}
```

Figure 2: 3APL Agent example

### 3.1 Agent mental components

As stated above, both 3APL and BDI-Delp are based on the BDI agent theory. Agents specified using the BDI theory are described by three mental components: Beliefs, Desires and Intentions. Thus the analysis in this section will be divided in three parts, one for each mental component of the BDI theory.

#### *3.1.1 Beliefs*

In the BDI-DeLP proposal, agent beliefs are specified by a DeLP program called $\mathcal{P}_B = (\Pi_B, \Delta_B)$. In $\Pi_B$ two disjoint subsets will be distinguished: $\Phi$ of perceived beliefs that will be updated dynamically, as the agent perceives the environment, and $\Sigma$ of strict rules and facts that represent static knowledge. Thus $\Pi_B = \Phi \cup \Sigma$. The agent of Fig.1 has $\Sigma = \emptyset$. Besides the perceived beliefs, the agent may use strict and defeasible rules from $\mathcal{P}_B$ to obtain a warrant for its derived beliefs. The facts of $\Sigma$ are not perceived, they do represent agent features, roles, etc. Therefore an agent in this proposal will have different types of beliefs:

- ***Perceived belief***:it is a fact of $\Phi$ that the agent has perceived directly from its environment.
- ***Strict belief***: it is a literal that is not a perceived belief, and its derived only from $\Pi_B$.
- ***Defeasible belief***: it is a warranted literal L supported by a non empty argument $\mathcal{A}$
- ***Derived belief***: it is a strict or a defeasible belief.

In Fig.1 the $BR$ are the rules used to get the ***Derived beliefs***.

**Example 2** *The **Derived beliefs** of the agent of Fig.1 will be* $\{b_2\}$ *, because there is an undefeated argument* $(b_2 \text{—}\prec s_3)$ *for literal* $b_2$*. Thus, this agent will believe in* $\{s_1, s_3, s_4, s_5, b_2\}$

The BDI-DeLP does not specify any mechanism to manipulate beliefs. However two processes can be identified: one to update the ***Perceived beliefs*** in every deliberative cycle; and other to add, modify or remove beliefs rules from $\mathcal{P}_B$, when the agent decides to do so.

In 3APL agent beliefs can be specified by any logic language [5]. However, as it was exposed in Section 2.2 3APL is an agent programming language, so a logic language for beliefs must be given. In their works they use propositional logic or Prolog logic. In this work we will study the approach of 3APL that uses Prolog to specify beliefs (see 3APL platform [9]). Thus, agent beliefs are specified by a Prolog program called `BELIEF BASE`. Therefore, a Belief can be a fact or a literal derived by Prolog rules. The agent can consult about its beliefs using a Prolog query.

In 3APL beliefs are not categorized. However the categorization exposed in BDI-DeLP above can be partially emulated. To specify ***Perceived beliefs***, Prolog facts can be used. This can be seen

in Fig.2 with the fact `perception([s1, s3, s4, s5])`, where the argument is the list of the perceived facts. ***Strict beliefs*** can be specified using Prolog rules. Using a Prolog program is not possible to specify ***Defeasible beliefs***, because Prolog does not have defeasible rules. Hence the ***Derived beliefs*** are only strict beliefs. Defeasible rules allow the programmer, for example to express easily exceptions of general rules (see [4] for examples).

It can be seen in Fig.1 and Fig. 2 that the representation of the agent of Ex.1 beliefs is different. One is made in Prolog and the other in DeLP. Form this difference, some issues arise. First, DeLP allows four types of answers for a query, whereas Prolog allows only two. Thus some kind of decision choice must be made in the Prolog approach, at the cost of losing flexibility in the agent reasoning process. It can be seen in the example of Fig.2 that the decision choice taken in this work is that UNDECIDED and UNKNOWN answers will be NO in Prolog. Nevertheless, this decision choice is not as restrictive as it seems, because it is considered reasonable to be skeptical about Beliefs. Second, a DeLP Program can sometimes be emulated by Prolog program. However, this emulated DeLP program will have more complex rules, will lose declarativity and will be less scalable. This comes because the Prolog rules will try to emulate the argumentative process done in DeLP. This can be seen in the form of the belief rules related to $b1$ in Fig.1 and Fig.2. Moreover, If the selection criterion of DeLP is changed or more rules related to $b1$ are added to $BR$ in Fig.1, all the rules related to `b1` in the `BELIEF BASE` of Fig.2 should be revised and in the worst case all will be changed.

**Example 3** *Consider the agents of Fig.1 and Fig.2. If the belief rule $\sim b_1 -\prec s_3$ is added to the $Beliefs\ Rules$ set of Fig.1, then the term `not(s3)` should be added to all rules related to `b1` in `BELIEF BASE` of Fig.2.*

Third, DeLP has mechanism to specify and use Strong Negation and Prolog does not. In brief, most of these issues come from the advantages DeLP over Prolog as knowledge representation languages. These advantages are exposed in [4].

3APL, as an programming language needs an explicit mechanism to add, remove or revise beliefs. For this, 3APL provides the set `CAPABILITIES`, which contains rules to add or remove beliefs. These rules, are called *basic actions* and can be used as a part of a plan. However, as it was said above, 3APL does not categorize beliefs. Therefore, all the beliefs are treated equally, which means that all the beliefs are updated using *basic action* rules. This means that the agent will update its perception beliefs only when *basic action* rules are executed, that is when it is executing a plan. Moreover, if the agent wants to update its perception beliefs every deliberation cycle, every plan it executes should include the *basic action* rules that update the perception (plans are explained in Section 3.1.3). This can be seen in Fig.2 where every rule in the `RULEBASE` set has, as part of its plan, the sequence `Sense(X),Add(X)`. Where `Sense(X)` is an external function to sense the environment and `Add(X)` is the basic action to add the acquired information `X` to the `BELIEF BASE`

### *3.1.2 Desires*

In BDI-DeLP the set $D$ is used to specify desires of the BDI theory. This set is composed of Literals, and can be contradictory. It represents all the desires that the agent want to achieve. A desire in this proposal represents an action that the agent wants to do. The agent only adopts a subset of desires, each time. This subset contains the current achievable desires and will be called $\mathbf{D}^c$. To do this a DeLP program $\mathcal{P}_F = (\Pi_F, \Delta_F)$ called *Filtering Rules* is used. This program $\mathcal{P}_F$ contains strict and defeasible rules that represent reasons for and against adopting desires (see Fig.1). To build the set $\mathbf{D}^c$, the DeLP program $K_{ag}$= $(\Pi_{ag}, \Delta_{ag})$ is used. The program $K_{ag}$ represents the agent knowledge base and it is created merging $\mathcal{P}_F$ and $\mathcal{P}_B$ (see [7] for details).

**Example 4** *Consider the sets* $\Phi$, $BR$ *and* $\mathcal{P}_F$ *from Fig.1. The program* $\mathrm{K}_{ag}$= $(\Pi_{ag}, \Delta_{ag})$ *will be*

$$\Pi_{ag} = \left\{ \begin{array}{cc} s_1 & s_3 \\ s_4 & s_5 \end{array} \right\} \quad \Delta_{ag} = \left\{ \begin{array}{llll} b_1 -\!\prec s_1 & b_1 -\!\prec s_2 & \sim b_1 -\!\prec s_5 & \sim b_1 -\!\prec s_1, s_2 \\ b_2 -\!\prec s_3 & \sim b_2 -\!\prec s_2, s_4 & g_1 -\!\prec b_1 & \sim g_1 -\!\prec b_2 \\ g_2 -\!\prec s_3 & \sim g_2 -\!\prec \sim b_1 & & \end{array} \right\}$$

In Ex.4 the $K_{ag}$ program is composed by all rules from $BR$ and $\mathcal{P}_F$, and all the literals in $\Phi$ of Fig.1. Besides $K_{ag}$ a *selection criterion T* is used. This criterion establishes the answers of $K_{ag}$ that will be acceptable for the agent. Therefore, it determines the type of the agent. The elements in $\mathbf{D}^c$ will be only those elements of $D$ that satisfy the selection criterion *T* in $K_{ag}$. To this purpose a filtering function is used. This function takes the selection criterion *T* and the desire set $D$, and builds $\mathbf{D}^c$. In [6] two possible selection criterion T are specified: one for *cautious agents*, that accepts only desires $\delta$, if there is a warrant for $\delta$ from $K_{ag}$; and the other for *bold agents*, that accepts desires $\delta$, if there is no warrant for $\sim\delta$ from $K_{ag}$. In the example of Fig.1 the agent type is not shown, but we assume that the agent is cautious, so he believes and desires only warranted things.

**Example 5** *Consider a cautious agent A, with* $\mathrm{K}_{ag}$ *of the Ex. 4, and* $D$ *of Fig.1. The set* $\mathbf{D}^c$ *set will be* $\{g_2\}$*, because there is an undefeated argument for* $g_2$ ($g_2 -\!\prec s_3$)*, and there is no warrant for* $g_1$.

In BDI-DeLP there is no mechanism to drop, modify or add new desires to the desire set D. However, the agent can have all the desires that will use in his lifespan in the set D, and use the $K_{ag}$ to select which of them are currently are achievable. We assume that a agent can not adopt new desires. Thus the agent selects desires from the desire pool that are currently achievable under some policy.

In 3APL the `GOAL BASE` set is used to specify Desires of the BDI theory. This set is composed by terms, which are called goals. Goals represent the situation where the agent wants to be in. The goals in `GOAL BASE` are those goals that the agent want to achieve. Thus the `GOAL BASE` is similar to the desire set $D$ of BDI-DeLP, which is exposed above. In 3APL there is no independent mechanism to determine which goals are currently achievable. Hence, this means that all goals in the `GOAL BASE` can be adopted as intentions.

In our opinion the agent development tool should provide some way to specify conditions over the desires. In the BDI-DeLP approach this is done via the filtering rules explained above. In 3APL, there are three alternatives for doing it, two indirectly via the *reasoning rules* (see 3.1.3), and the other at a meta level with the deliberative cycle program. One of those alternatives is using the goal rules (see 3.1.3), where the agent can manipulate the `GOAL BASE` to emulate the currently achievable desires set $\mathbf{D}^c$ of the other proposal. For us, this alternative leads the agent to have desires to change other desires. Other alternative is putting every condition for a goal as part of the precondition of the *reasoning rules* related to that goal. This is the alternative that was adopted in Fig.2. There, it can be seen that both reasoning rules related to the goal `g1` have, as part of their preconditions, the consults `b1, not(b2)`. This tries to emulate the conditions that are established by the filtering rules $\{\ g_1 -\!\prec b_1\ ,\ \sim g_1 -\!\prec b_2\ \}$ of the example from Fig.1. This design choice leads to some issues over the *reasoning rules* that will be detailed in the next section (because they are related to the intention model). Finally, the last alternative to expose conditions over goals in 3APL is using the tools to change the deliberative cycle program, which are proposed in [2]. These tools allow to redefine the deliberative cycle of the 3APL interpreter. In the redefinition the developer can put conditions related to the goals and the beliefs in the deliberative cycle. Thus the behavior of the filtering rules of the other approach can be emulated using these tools. These tools are at a meta level of the agent (The interpreter algorithm is changed) thus, every time a rule over the desires is to be added, the whole deliberative cycle algorithm must be revised. Therefore, compered to the other ways to specify the conditions over desires in 3APL this way is more unscalable. Thus, using one of these alternatives the filtering rules of BDI-DeLP can be emulated in 3APL. However, the same issues marked in Section 3.1.1 and exemplified in Ex.3 will arise.

In 3APL it is possible to define the agent type using the tools to redefine the delineative cycle program[2], exposing conditions over the execution of the mental components.

### *3.1.3 Intentions*

In BDI-DeLP *intention rules* are used to specify how desires are adopted as intentions. An intention in this model is a current desire that the agent can commit with under some established preconditions and constrains. An *intention rule* will be applicable if its associated desire is in the $\mathbf{D}^c$ set and its preconditions and constrains are satisfied. Satisfied preconditions are warranted beliefs of $K_{ag}$ or desires in $\mathbf{D}^c$. Satisfied constrains are not warranted beliefs of $K_{ag}$ or desires that are not in $\mathbf{D}^c$. To select between various applicable intention rules, a selection policy is used. This policy can be specified by the developer. In the example of Fig.1 the intention rules are model by the set $IR$, and the policy used is *"Select the first applicable rule"*.

**Example 6** *Consider the agent of Fig.1,* $\mathrm{K}_{ag}$ *of Ex.4 and* $\boldsymbol{D}^c$ *of Ex.5. The only applicable rule from* $IR$ *of this agent will be* $g_2 \Leftarrow \{s_4\}\{\}$*, because* $g_2$ *is the only desire in the set* $\boldsymbol{D}^c$ *and* $s_4$ *is derivable from the program* $\mathrm{K}_{ag}$*.*

The BDI-DeLP proposal does not specify a way to manipulate or write the actions associated with a selected intention, which is left to future works. Also in their approach there is no mechanism to add, modify or remove Intention Rules.

In 3APL intentions are specified by the set of rules called `RULEBASE`. The rules of this set are called *Reasoning Rules* and can be divided in three types: the *interaction rules*, the *goal rules* and the *plan rules*. The *plan rules* are used to revise or drop plans, and will not be treated in this work. The *interactions rules* and the *goal rules* are used to specify intentions.

*Interaction rules* of 3APL allow the agent to reach its goals via plans. These rules consist of a header goal, a guard, and a plan. The header goal is the goal which the agent will commit to. The guard represents the preconditions that must be satisfied (with respect to the `BELIEF BASE`) to apply the rule. The plan represents the set of actions that the agent will do to achieve the selected goal. Thus a rule of this set is a map under some established preconditions between a goal and a plan to solve it. The *goal rules* are very similar, in its form, to the interaction rules. The difference between these rules types is that the plans of the goal rules are only actions over the `GOAL BASE`. In the example of Fig.2 the set `RULEBASE` is only composed by *interaction rules* (No *goal* or *plan rules* were used).

In 3APL a interation/goal rule is applicable if its header goal is in the `GOAL BASE` and its guard is satisfied. To satisfy a guard all its elements must be derivable from the `BELIEF BASE` Prolog program. Once a rule is selected, its plan is executed.

**Example 7** *Consider the 3APL agent of Fig.2. The only applicable interaction rule is* `g2 <- perceived(s4),perceived(s3),not(tail1)|{PlanGoal2,Sense(X),Add(X)}` *because g2 is in the* `GOAL BASE` *and the guard elements of the guard* `perceived(s4)`, `perceived(s3)` *and* `not(tail1)` *are derivable from the* `BELIEF BASE` *program. Note that, despite* `g1` *is in the* `GOAL BASE`*, all the interaction rules related to* `g1` *do not satisfy its guards.*

To select between two or more applicable rules, a policy can be used. In the 3APL Platform [9] a simple policy is used: Select the first specified rule (The same that we used in the example of BDI-DeLP). However, this policy can be redefined using the tools to redefine the deliberative cycle [2].

As it was mentioned above the guard of a 3APL interaction/goal rule will have all preconditions related to the applicability of an intention. However, it was exposed in 3.1.2 that the conditions over goals are added to the guard of the interaction/goal rule. Thus, the guard of the interaction/goal rule in 3APL will have preconditions to verify two semantically different things: if the rule is applicable; and if the associated goal is achievable. This issue add some undesired practical effects in the rules.

**Example 8** *Consider the agent of Fig.2. It can be seen that* `b1, not(b2)`*, which represents the conditions for the* `g1`*, is repeated in every rule that has* `g1` *as a header goal.*

In general, all the conditions over a goal will be present in the guard of every interaction/goal rule related to that goal. This "repeated code problem" can become bigger if the conditions over the goals are more complex. These issues are not present in BDI-DeLP, because it provides different mechanisms to represent conditions over desires and preconditions over intention rules.

Contrary to BDI-DeLP, 3APL, as agent programming language has a well defined model to specify the actions (plans) associated with the intention, and rules (*plan rules*) to revise them. 3APL does not specify a mechanism to add, modify or remove reasoning rules.

### 3.2 Reasoning

In this section the deliberative cycle and the reasoning processes of both approaches are analyzed.

The BDI-DeLP approach does not explicitly define a deliberative cycle, it is left open for the developer. However, the deliberative cycle can not be arbitrary defined. There are several restrictions on how the cycle can be built. These restrictions are due to the way the sets for the reasoning process are used. Next, one possible deliberative cycle will be exposed:

1. Find which desires are currently achievable, using the $K_{ag}$ DeLP program (beliefs and the filtering rules).
2. Find intention rules that match with the desires selected in step 1.
3. Find which rules, of those selected in step 2, satisfy the preconditions and constrains, using the $K_{ag}$ DeLP program
4. If one or more rule satisfies step 3, then select one rule using the policy p.

The reasoning process of the BDI-DeLP proposal has two stages. One when the agent determines if a desire is in the set of the current desires, and the other when it checks if an intention rule is applicable. The applicable intention policy can also be thought as part of the reasoning process.

3APL, as an agent programming language has a concrete deliberative cycle specified [9],[2]. However, as it was stated in previous sections there are tools to reprogram the procedure that specifies the cycle [2]. The 3APL Deliberative cycle exposed in[2] is described next:

1. Find a reasoning rule matching with goals in the `GOAL BASE`.
2. Find which rules of step 1 satisfy its preconditions (Guard) with respect to the `BELIEF BASE` Prolog program.
3. If one or more rules satisfy steps 1 y 2, select a rule and execute its plan.

The reasoning process of 3APL has one stage, that is, when the agent checks if a rule is applicable or not. Considering what was stated in Section 3.1.3, this means that 3APL will test in the same reasoning step if a goal is achievable and if an interaction/goal rule is applicable. This leads to a practical issue. It was explained in Section 3.1.2 and 3.1.3, that all conditions for a goal will be in the guard of a rule that has that goal as a header goal. Thus the conditions for a goal will be tested in every rule that has that goal as a header goal. For example, it can be seen in Fig.2 that the conditions `b1, not(b2)` of the goal `g1` are not satisfied (`b1` is not derived from the `BELIEF BASE` program). These conditions will be tested with the guard of every rule related to `g1`. This means that these conditions will be tested two times in every deliberative cycle. This problem is not present in the BDI-DeLP approach because it does desire filtering and intention rule applicability tests in two separate steps. Thus, if 3APL would have two steps for those two stages the problem exposed above would not be present.

# 4 3APL WITH DELP AS A BELIEF LANGUAGE: 3APL-DELP

In the last section an analysis of each approach was shown and there, strong points of each approach were highlighted. In this section some of these strong points are used to present and define a combined approach called 3APL-DeLP.

In the presentation of 3APL [5], the authors did not fix the logic language used for knowledge representation. To develop the rest of their works [2], [10], [3] they used propositional logic or Prolog as language to represent beliefs. In particular the 3APL interpreter[9] uses Prolog. In this work we propose a to use DeLP as the knowledge representation language in 3ALP. This will be done using a DeLP program as the `BELIEF BASE` program of 3APL. This proposal will be called 3APL-DeLP.

A 3APL-DeLP agent will have the same mental components as a 3APL agent. Thus the mental state (configuration) of 3APL-DeLP is the same given in[3]. Next, we will define the components of 3APL that are affected by our proposal.

It was shown in Section 3, that DeLP is a more sophisticated mechanism for knowledge representation than Prolog. Thus, a the program used to represent beliefs of 3APL-DeLP must be defined as follows

**Definition 1** : ***(*Belief Base*)*** *The `BELIEF BASE` of a 3APL-DeLP Agent will be a DeLP program $\mathcal{P}_{BF} = (\Pi_{BF}, \Delta_{BF})$, where $\Pi_{BF}$ is a non contradictory set of facts and $\Delta_{BF}$ is a set of defeasible rules.*

All the Beliefs that the agent believes are derivable by the `BELIEF BASE` program, therefore:

**Definition 2** : ***(*Belief*)*** *Let $\mathcal{P}_{BF}$ a be `BELIEF BASE`, a Literal $\mathcal{B}$ is a Belief from $\mathcal{P}_{BF}$ iff $\mathcal{B}$ is warranted from $\mathcal{P}_{BF}$.*

In 3APL basic action rules can be used to add or remove beliefs from the Belief Program. Therefore:

**Definition 3** : ***(*Basic Action Rule*)*** *A Basic Action rule $Ba$ is an ordered triplet $Ba$=($Pre$, $Name$, $Pos$), where: $Pre$=$\{P_1, \ldots, P_m\}$ is a set of Literals representing a preconditions for $Ba$, $Name$ is a predicate, and $Pos$=$\{X_1, \ldots, X_n\}$ is a consistent set of extended Literals representing the effects of executing $Ba$.*

All the basic action rules are in the `CAPABILITIES` set of the 3APL-DeLP agent. The predicate $Name$ of Def.3, is used to call capability rule from the plans. This predicate can contain variables which will be treated as in-mode parameters.

**Definition 4** : ***(*Applicable Basic Action Rule*)*** *Let $Ba$ be a Basic Action Rule in `CAPABILITIES` and $\mathcal{P}_{BF}$ a be `BELIEF BASE`. $Ba$ will be applicable if every $P_i$ in $Pre$ is a Belief.*

**Definition 5** : ***(*Basic Action Rule Effect*)*** *Let $Ba$ be an Applicable Basic Action Rule in CAPABILITIES and $\mathcal{P}_{BF}$ a be `BELIEF BASE`. The effect of executing $Ba$ is is the revision of the `BELIEF BASE` DeLP program. This revision will consist of removing any literal in $\mathcal{P}_{BF}$ that is complementary of any literal in $Pos$ or that is preceded by a "not" in $Pos$, and then adding all the literals in $Pos$ to the resulting program.*

In 3APL an Interaction Rule of the `RULEBASE`, as it was explained in Section 3.1.3, involve a goal, a guard and a plans. The plan is the specifies the actions needed to solve the goal and the guard are the preconditions of the rule. Those preconditions are queries to the `BELIEF BASE`. Therefore:

**Definition 6** : ***(*Interaction Rule*)*** *A Interaction rule $R$ is an ordered triplet $R$=($HG$, $Guard$, $Pl$), where: $HG$ is a Literal representing the goal to achieve, $Guard$=$\{G_1, \ldots, G_n\}$ is a consistent set of extended Literals representing the preconditions for $R$, and $Pl$ is a plan.*

Note that the definition of *goal rules* is analogous to Def.6, there, a goal action should be in the place of the plan.

**Definition 7** : ***(*Applicable Interaction Rule*)*** *Let $R$ be a Interaction rule in `RULEBASE` and $\mathcal{P}_{BF}$ a be `BELIEF BASE`. $R$ will be applicable if $HG$ is in `GOALBASE` and each $G_i$ in $Guard$ is a Belief.*

Thus, using DeLP as the BELIEF BASE program allows to model the rules to represent knowledge in a more declarative way. DeLP characteristics allows to model incomplete and potentially contradictory information. The inference mechanism of DeLP allows the 3APL-DeLP agent to decide between several contradictory conclusions, and to adapt to changing environments, *i.e.*, it allows to add or remove information in a dynamic way, without the need of changing every rule of the agent, which adds scalability and flexibility. More examples of DeLP as a knowledge representation language are shown in [4]. Next, in Fig.3, a complete specification of the agent of Ex.1 is shown.

```
CAPABILITIES {
    {perception(Y)} Add(X) {not perception(Y), perception(X)}
} BELIEFBASE {
   perception([s1, s3, s4, s5]),
   perceived(X) -< perception(L),member(X,L).
   b1 -< perceived(s1).
   b1 -< perceived(s2).
   ~b1 -< perceived(s5).
   ~b1 -< perceived(s1),perceived(s2).
   b2 -< perceived(s3).
   ~b2 -< perceived(s2),perceived(s4).
} GOALBASE {
    g1, g2
} RULEBASE {
    g1 <- perceived(s1), perceived(s4), b1, not(b2) | {Plan1Goal1, Sense(X), Add(X)},
    g1 <- perceived(s4), b1, not(b2) | {Plan2Goal1, Sense(X), Add(X)},
    g2 <- perceived(s4), perceived(s3), not(~b1) | {PlanGoal2, Sense(X), Add(X)},}
```

Figure 3: 3APL-DeLP Agent example

As shown in Fig.3 the BELIEF BASE is a DeLP program is similar of the $BR$ and $\Phi$ of Fig.1. The only difference between these two DeLP programs is that perceptions in Fig.3 were specified in the same way of the 3APL agent of Fig.2. It can be seen that new rules can be added without the need of changing all the BELIEF BASE program. Thus 3APL-DeLP will allow to specify BDI based agents with the benefits of the declarativity, planning support and communicative support of 3APL and all the benefits of DeLP as a knowledge representation language.

# 5 CONCLUSIONS AND FUTURE WORK

In this work we have shown 3APL-DeLP which is an approach to a programming Language for BDI Agents that uses defeasible argumentation for agent knowledge representation and reasoning. To reach this programming language two approaches based on BDI theories were studied. 3APL a programming language for cognitive agents and BDI-DeLP an BDI agent architecture that uses DeLP for knowledge representation and reasoning.

First, an introduction for each approach was given. There, goal, scope and advantages of each were characterized, and an example of an agent specified in each approach was shown. Then an analysis of the knowledge representation and reasoning features of each approach was developed. This analysis was divided in the Belief, Desires, Intention and reasoning of each approach. There, the advantages of DeLP as knowledge representation were characterized and shown with examples.

Finally, considering the advantages of DeLP exposed in the analysis of both approaches, the advantages of 3APL as agent programming language and the fact that in 3APL the knowledge representation language is not fixed, 3APL-DeLP was proposed. In 3APL-DeLP the agent knowledge is represented by a DeLP-program and the agent may reason with a defeasible argumentation formalism. Definitions and an example for this programming language were given. Thus, 3APL-DeLP allows the

specification of BDI based agents with the benefits of the declarativity, planning support and communicative support of 3APL and all the benefits of DeLP as a knowledge representation language.

In this work we only gave the definition of those components of 3APL that were affected by our proposal, however a full revision of other components is needed. Also, further research in how planning is affected by our proposal should be done. Finally, we think that filtering rules can be added to 3APL-DeLP, this has been left for future research.

## REFERENCES

[1] M. E. Bratman, D. Israel, and M. Pollack. Plans and resource-bounded practical reasoning. In Robert Cummins and John L. Pollock, editors, *Philosophy and AI: Essays at the Interface*, pages 1–22, Cambridge, Massachusetts, 1991. The MIT Press.

[2] M. Dastani, F. de Boer, F. Dignum, and J. Meyer. Programming agent deliberation: An approach illustrated using the 3apl language. In *The Second Conference on Autonomous Agents and Multi-agent Systems (AAMAS'03), Melbourne, Australia*, 2003.

[3] M. Dastani, B. van Riemsdijk, F. Dignum, and J. Meyer. A programming language for cognitive agents: Goal-directed 3apl. In *First Workshop on Programming Multiagent Systems: Languages, frameworks, techniques, and tools (ProMAS03), Melbourne, Australia*, 2003.

[4] A. Garcia and G. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(1-2):95–138, 2004.

[5] K. Hindriks, F. de Boer, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming in 3apl. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.

[6] N. D. Rotstein, A. J. García, and G. R. Simari. Reasoning from desires to intentions: A dialectical framework. In *22nd. AAAI Conference on Artificial Intelligence*, July 2007.

[7] N. D. Rotstein, A. J. García, and Guillermo R. Simari. Defeasible reasoning about beliefs and desires. In *11th International Workshop on Non-Monotonic Reasoning (NMR 2006)*, pages 429–436. Lake District, UK, May 2006.

[8] Y. Shoham. Agent-oriented programming. In *Artificial Intelligence*, 1993.

[9] E.C. ten Hoeve. 3apl platform. Master's thesis, Univ. of Utrech, 2003.

[10] J. van der Ham. Multi-agent fipa compliant 3apl agents. Master's thesis, Univ. of Utrech, 2002.

[11] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.