Adding Knowledge Updates to 3APL

Vivek Nigam* and João Leite

CENTRIA, New University of Lisbon,Portugal
vivek.nigam@gmail.com, jleite@di.fct.unl.pt

Abstract. 3APL is a widely known multi-agent programming language. However, when to be used in certain domains and environments, 3APL has some limitations related to its simplistic update operator that only allows for updates to the extensional part of the belief base and its lack of a language with both default and strong negation to enable the representation and reasoning about knowledge with the open and closed world assumptions. In this paper, we propose to address these issues by modifying the belief base of 3APL to be represented by Dynamic Logic Programming, an extension of Answer-Set Programming that allows for the representation of knowledge that changes with time.

1 Introduction

In the past few years, several agent architectures and agent programming languages have been proposed. Among them we can find, for example, 3APL [9,12], FLUX [22], IMPACT [10], DALI [8], JASON [5] and Minerva [14,18]. For a survey on some of these systems, as well as others, see [6,7,20].

In this paper, we take a closer look at 3APL, one of the existing systems that has recently received an increasing amount of attention, and propose some enhancements to its language and semantics.

3APL is a logic based programming language for implementing cognitive agents that follows the classical BDI architecture where agents have beliefs (B), intentions (I) and desires (D) to guide their actions. The semantics of 3APL agents is defined by a *transition system* composed of *transition rules*. The use of 3APL provides the agent programmer with a very *intuitive* and simple way to define agents. The programmer can declaratively specify the *beliefs* (represented by Horn Clauses) and *goals* (represented by conjunctions of atoms) of agents, how they *build plans* to achieve such goals, and reason with their beliefs. Furthermore, communication between agents can be done in an elegant way by modifying the beliefs of agents, allowing for the possibility of reasoning with the transferred messages. Despite all these interesting properties, 3APL, when to be used in certain domains and environments, has some limitations that serve as our motivation to propose the modifications presented in this paper. These limitations, in our opinion, are:

* Supported by the Alβan Program, the European Union Programme of High Level Scholarships for Latin America, no. E04M040321BR.

R.H. Bordini et al. (Eds.): ProMAS 2006, LNAI 4411, pp. 165–181, 2007.

1. Limited belief updates - The mechanism used by 3APL to update agent's beliefs is quite limited. Such updates in 3APL amount to the simple addition and removal of facts in the agent's belief base. It is not difficult to find a situation where this type of belief update is insufficient. Consider an agent with a belief base containing the rule $believe(santa\_claus) \leftarrow mother\_said(santa\_claus)$, and the fact $mother\_said(santa\_claus)$. This agent can be seen as a child agent that believes in everything its mother says, in this case it believes in *santa claus*, because its mother said so ($mother\_said(santa\_claus)$). Furthermore, consider that the agent evolves and discovers that in fact, *santa claus* doesn't exist, even though its mother said so. Since 3APL only allows for updates to the extensional part of the belief base (i.e. its set of facts), it is not possible to achieve the desired semantics, where $believe(santa\_claus)$ is false and $mother\_said(santa\_claus)$ is true, by the mere addition and retraction of facts. Note that it is not possible to remove the fact $believe(santa\_claus)$ because there is none to be removed, and if the fact $mother\_said(santa\_claus)$ is removed it would be change the belief base in an undesired way. To obtain the desired effect, updates in the intensional part of the knowledge base (i.e. its set of rules) are required;

2. Limited expressive power of negative information - 3APL allows for the use of one form of negation, namely *negation by finite failure*. It has been shown that the use of default negation (*not*) provides good expressive power to a language. Furthermore, the use of both default and strong negations ($\neg$), concurrently, such as in Answer-Set Programming [11], allows for easy ways to reason with both the *closed* and *open world assumptions*. For example, in the classical car-train cross, where the car should pass the cross if its sure that the train is not coming, it is necessary to reason with the open world assumption, where strong negation plays a key role ($\neg train$). On the other hand, to represent a cautious agent that would move if it believes that a place is not safe ($not\ safe$), the use of default negation is more adequate;

In this paper, we will use Dynamic Logic Programming (DLP) [19,2,14], an extension of Answer Set Programming, to address these limitations stated above. We propose to represent the 3APL agent's *belief base* as a Dynamic Logic Program.

According to the paradigm of *DLP*, knowledge is given by a series of theories, encoded as generalized logic programs[1], each representing distinct states of the world. Different states, sequentially ordered, can represent different time periods, thus allowing DLP to represent knowledge that undergoes successive updates. Since individual theories may comprise mutually contradictory as well as overlapping information, the role of *DLP* is to employ the mutual relationships among different states to determine the declarative semantics for the combined theory comprised of all individual theories at each state. Intuitively, one can add, at the end of the sequence, newer rules (arising from new or reacquired knowledge) leaving to *DLP* the task of ensuring that these rules are in force, and that previous ones are valid (by inertia) only so far as possible, i.e. that

[1] Logic programs with default and strong negation both in the body and head of rules.

they are kept for as long as they are not in conflict with newly added ones, these always prevailing.

By using DLP to represent the agent's *belief base*, we address, at once, both of the limitations stated above. The first, namely the one related to the scope of the existing 3APL update operator, is immediately solved by the very foundational scope of DLP, after 3APL is adapted to accommodate such change. With DLP, 3APL agents will be able to maintain an *up to date belief base* in situations where both the extensional and intensional parts of the knowledge base change. Agent's simply have to add, at the end of the sequence of programs that constitutes their *belief base*, new facts and rules alike, and not worry with emerging contradictions with previous rules as the DLP semantics properly handles them. The second limitation is also addressed by using DLP, as the object language used to define the generalized logic programs allows for both default and strong negations, inherited from Answer-Set Programming [11] that it generalizes.

En passant, we take the opportunity provided by the fact that DLP allows for rule based updates, to also increase the expressiveness of the messages transmitted between the agents, by allowing their content to consist of generalized logic programs. By transmitting logic programs, instead of facts, agents will be able to exchange knowledge containing rules. Depending on its beliefs, the receiving agent can update its beliefs by the transmitted logic program, thus facilitating learning (through rule teaching).

This remainder of the paper is distributed in the following way. We begin in the Section 2 to give some preliminary definitions related to 3APL and Dynamic Logic Programming that will be used throughout the paper. Later, in Section 3, we modify the syntax of some of the transition rules of 3APL. In Section 4 we present the semantics of the belief query language and of the proposed transition rules. In Section 5, we discuss some of the added features obtained by the modification proposed and in Section 6 we give an illustrative example with some of the properties of the modified system. Finally, in Section 7 we conclude with some suggestions of further investigation.

2 Preliminaries

In this Section, after introducing some concepts of *logic programs*, we introduce the semantics of *Dynamic Logic Programs* and *partially* introduce the *3APL multi agent language* in its propositional form. For the sake of space, we are only going to introduce the reader the definitions of 3APL that are *relevant* for this paper, further details about the complete version of 3APL system can be found in [9].

2.1 Languages and Logic Programs

Let $\mathcal{K}$ be a set of propositional atoms. An *objective literal* is either an atom A or a strongly negated atom $\neg A$. A *default literal* is an objective literal preceded by *not*. A *literal* is either an objective literal or a default literal. We also define

the set of objective literals $\mathcal{L}_{\mathcal{K}}^{\neg} = \mathcal{K} \cup \{\neg A \mid A \in \mathcal{K}\}$ and the set of literals $\mathcal{L}_{\mathcal{K}}^{\neg,not} = \mathcal{L}_{\mathcal{K}}^{\neg} \cup \{not\ L \mid L \in \mathcal{L}_{\mathcal{K}}^{\neg}\}$ over the alphabet $\mathcal{K}$. We are going to use the set *Disjunction* to build the belief query language, $\mathcal{L}_B$, if $A \in \mathcal{K}$ then $B(A), \neg B(A) \in$ *Disjunction*, $\top \in$ *Disjunction* and if $\delta, \delta' \in$ *Disjunction* then $\delta \vee \delta' \in$ *Disjunction*, if $\delta \in$ *Disjunction* then $\delta \in \mathcal{L}_B$, furthermore if $\phi, \phi' \in \mathcal{L}_B$ then $\phi \wedge \phi' \in \mathcal{L}_B$. Informally, $\mathcal{L}_B$ is the smallest set containing all the formulas in conjunction normal form, where $B(.)$ and $\neg B(.)$ are the *literals* of the language. The goal query language, $\mathcal{L}_G$, is defined the following way, $\top \in \mathcal{L}_G$, if $A \in \mathcal{K}$ then $G(\phi) \in \mathcal{L}_G$, and if $k, k' \in \mathcal{L}_G$ then $k \wedge k' \in \mathcal{L}_G$.

A *rule* r is an ordered pair $Head\,(r) \leftarrow Body\,(r)$ where $Head\,(r)$ (dubbed the head of the rule) is a literal and $Body\,(r)$ (dubbed the body of the rule) is a finite set of literals. A rule with $Head\,(r) = L_0$ and $Body\,(r) = \{L_1, \ldots, L_n\}$ will simply be written as $L_0 \leftarrow L_1, \ldots, L_n$. A *generalized logic program* (*GLP*) P, in $\mathcal{K}$, is a finite or infinite set of rules. If $Head(r) = A$ (resp. $Head(r) = not\ A$) then $not\ Head(r) = not\ A$ (resp. $not\ Head(r) = A$). If $Head\,(r) = \neg A$, then $\neg Head\,(r) = A$. By the *expanded generalized logic program* corresponding to the GLP P, denoted by $\mathbf{P}$, we mean the GLP obtained by augmenting P with a rule of the form $not\ \neg Head\,(r) \leftarrow Body\,(r)$ for every rule, in P, of the form $Head\,(r) \leftarrow Body\,(r)$, where $Head\,(r)$ is an objective literal[2]. Two rules r and r' are conflicting, denoted by $r \bowtie r'$, iff $Head(r) = not\ Head(r')$. An *interpretation* M of $\mathcal{K}$ is a set of objective literals that is consistent i.e., M does not contain both A and $\neg A$. An objective literal L is true in M, denoted by $M \vDash L$, iff $L \in M$, and false otherwise. A default literal $not\ L$ is true in M, denoted by $M \vDash not\ L$, iff $L \notin M$, and false otherwise. A set of literals B is true in M, denoted by $M \vDash B$, iff each literal in B is true in M. An interpretation M of $\mathcal{K}$ is an *answer set* of a GLP P iff $M' = least\,(\mathbf{P} \cup \{not\ A \mid A \notin M\})$, where $M' = M \cup \{not\_A \mid A \notin M\}$, A is an objective literal, and $least(.)$ denotes the least model of the definite program obtained from the argument program by replacing every default literal $not\ A$ by a new atom not_A. For notational convenience, we will no longer explicitly state the alphabet $\mathcal{K}$. We will consider the alphabet of the language at an instant, consisting precisely of all the propositional symbols that appear *explicitly* in the program at such instant. Therefore, the alphabet of a program may change if new propositional symbols are included in the program. Furthermore, as usual, we will consider all the variables appearing in the programs as a shorthand for the set of all its possible ground instantiations.

2.2 Dynamic Logic Programming

A *dynamic logic program* (*DLP*) is a sequence of generalized logic programs. Let $\mathcal{P} = (P_1, ..., P_s)$, $\mathcal{P}' = (P'_1, ..., P'_n)$ and $\mathcal{P}'' = (P''_1, ..., P''_s)$ be DLPs. We use $\rho\,(\mathcal{P})$ to denote the multiset of all rules appearing in the programs $\mathbf{P}_1, ..., \mathbf{P}_s$, and $(\mathcal{P}, \mathcal{P}')$ to denote $(P_1, ..., P_s, P'_1, ..., P'_n)$, and $(\mathcal{P}, P'_1)$ to denote $(P_1, ..., P_s, P'_1)$.

[2] Expanded programs are defined to appropriately deal with strong negation in updates. For more on this issue, the reader is invited to read [15,14]. In subsequent sections, and unless otherwise stated, we will always consider generalized logic programs to be in their expanded versions.

Each position, i, of sequence of programs that constitutes a DLP, represents a state of the world (for example different time periods), and the corresponding logic program in the sequence, P_i, contains some knowledge that is supposed to be true at this state. The role of Dynamic Logic Programming is to assign a semantics to the combination of these possibly contradictory programs, by using the mutual relationships existing between them. This is achieved by considering only the rules that are not conflicting with rules in a GLP that is in a position ahead in the sequence of programs. Intuitively, one could add a new GLP to the end of the sequence, representing a new update to the knowledge base, and let DLP solve, automatically, the possible contradictions originated by this new update.

Definition 1 (Semantics of DLP). *[14,1] Let $\mathcal{P} = (P_1, \ldots, P_s)$ be a dynamic logic program over language $\mathcal{K}$, A an objective literal, $\rho(\mathcal{P})$, M' and least(.) as before. An interpretation M is a stable model of $\mathcal{P}$ iff*

$$M' = least\left([\rho(\mathcal{P}) - Rej(M, \mathcal{P})] \cup Def(M, \mathcal{P})\right)$$

Where:

$$Def(M, \mathcal{P}) = \{not\, A \mid \nexists r \in \rho(\mathcal{P}), Head(r) = A, M \vDash Body(r)\}$$
$$Rej(M, \mathcal{P}) = \{r \mid r \in \mathbf{P}_i, \exists r' \in \mathbf{P}_j, i \leq j \leq s, r \bowtie r', M \vDash Body(r')\}$$

We can use DLPs to elegantly represent evolving knowledge base, since their semantics is defined by using the whole history of updates and by giving a higher priority to the newer information. We will illustrate how this is achieved in the following example.

Example 1. Consider a DLP, $\mathcal{P}$, that initially contains only the program P_1, with the intended meaning that: if the tv is on (tv_on) the agent will be watching the tv ($watch\_tv$); if the tv is off it will be sleeping ($sleep$); and that the tv is currently on.

$$\begin{aligned} P_1 : \; & sleep \leftarrow not\, tv\_on \\ & watch\_tv \leftarrow tv\_on \\ & tv\_on \leftarrow \end{aligned}$$

The DLP has as expected, only one stable model, namely $\{watch\_tv, tv\_on\}$, where the agent is watching tv and not sleeping.

Consider now that $\mathcal{P}$ is updated by the program P_2, stating that if there is a power failure ($power\_failure$) the tv cannot be on, and that currently there is a power failure.

$$\begin{aligned} P_2 : \; & not\, tv\_on \leftarrow power\_failure \\ & power\_failure \leftarrow \end{aligned}$$

Since the program P_2 is newer than the previous program P_1, the rule, $tv\_on \leftarrow$, will be rejected by the rule $not\, tv\_on \leftarrow power\_failure$. Thus obtaining the expected stable model $\{sleep, power\_failure\}$, where the agent is sleeping and

the tv is no longer on. Furthermore, consider one more update stating that the power failure ended.

$$P_3 : not\, power\_failure \leftarrow$$

Because of the update P_3, the rule $\{power\_failure \leftarrow\} \subset P_2$ is rejected and $power\_failure$ should not be considered as true. Therefore, the rule $\{tv\_on \leftarrow\} \subset P_1$ is no longer rejected, and again the agent will conclude that the tv is on and it did not fall asleep. As expected, the stable model of the updated program is once more $\{watch\_tv, tv\_on\}$.

Of course, due to the lack of interesting programs on the television, it might happen that we don't watch tv even if the tv is on. We can use a new update, P_4, to represent this situation:

$$\begin{aligned} P_4 : \; & not\, watch\_tv \leftarrow bad\_program \\ & good\_program \leftarrow not\, bad\_program \\ & bad\_program \leftarrow not\, good\_program \end{aligned}$$

With this new update the DLP will have two stable models, one considering that the tv show is good and the agent is watching the tv ($\{good\_program, watch\_tv, tv\_on\}$), and another that the program is bad and it is not watching the tv ($\{bad\_program, tv\_on\}$).

As illustrated in the example above, a DLP can have more than one stable model. But then how to deal with these stable models and how to represent the semantics of a DLP? This issue has been extensively discussed and three main approached are currently being considered [14]:

Skeptical - $\models_\cap$ According to this approach, the *intersection* of all stable models is used to determine the semantics of a DLP. As we are going to represent the beliefs of the agent as a DLP, this approach would be best suited for more skeptical agents, since they would only believe in a statement if all stable models (possible worlds) support this statement;

Credulous - $\models_\cup$ According to this approach, the *union* of all stable models is used to determine the semantics of a DLP. With this semantics, a DLP would consider as true all the objective literals that are true in one of its stable models;

Casuistic - $\models_\Omega$ According to this approach, one of the stable models is selected, possibly by a selection function Ω, to represent the semantics of the program. Since the stable models of a belief base can be seen as representations of possible worlds, an agent using this approach would commit to one of them to guide their actions.

We will denote by $SM(\mathcal{P})$ the set of all stable models of the DLP $\mathcal{P}$. Further details and motivations concerning DLPs and its semantics can be found in [14].

2.3 Propositional 3APL

The 3APL agent is composed of a *belief base* (σ) that represents how the world is for the agent, a *goal base* (γ) representing the set of states that the agent

wants the world to be, a set of *capabilities* (Cap) that represents the set of actions the agent can perform, a *plan base* (Π) representing the plans that the agent is performing to achieve specific goals, sets of *goal planning rules and plan revision rules* (PG, PR) that are used by the 3APL agent to build plans, and the *environment* (ξ) in which the agent is situated. The environment can be viewed as a set of facts.

The belief base of the agent is composed of a set of rules of the form, $(A \leftarrow A_1, \ldots, A_n)$, where $A_1, \ldots, A_n, A \in \mathcal{K}$. The goal base is composed by a set containing sets of atoms[3], $\{\Sigma_1, \ldots \Sigma_n \mid \Sigma_i \subseteq \mathcal{K}, 1 \leq i \leq n\}$. Each set contained in the goal base will represent a goal of the agent. For example in the classical block world problem, if the goal base of an agent contains the set of atoms $\{on(A, B), on(C, D)\}$, a goal of the agent would be to have the block A over the block B ($on(A, B)$), and simultaneously the block C over the block D ($on(C, D)$).

Plans can be composed of several types of actions (*communication action*, *mental action*, *external action*, *test action*, *composite plans*, etc). We are interested in the *mental actions* and *communication actions*. We will denote the empty plan as ϵ. We will not formally define the language of plans ($\mathcal{L}_{\mathcal{P}}$), since it will not be extensively used in this paper, more interested readers are invited to read [9].

Definition 2 (Mental Actions Specifications). *[9] Let $\beta \in \mathcal{L}_B$ be the precondition of the mental action, α be a mental action name, $Lit_B = \{B(\phi), \neg B(\phi) \mid \phi \in \mathcal{K}\}$ and $\beta' = \beta_1 \wedge, \ldots, \wedge \beta_n$ be the postcondition of the mental action, where $\beta_1, \ldots, \beta_n \in Lit_B$ Then, a mental action is a tuple $\langle \beta, \alpha, \beta' \rangle$, and Mact is the set of all mental actions.*

The communication actions are represented by the special predicate $Send(r, type, A)$, where r is the name of the agent the message is being sent to, *type* is the performative indicating the nature of the message and the message $A \in \mathcal{K}$.

The semantics of an agent in 3APL is given by *transition rules*. We will be concerned in this paper with two transition rules corresponding to the *mental* and the *communication actions*.

Since the set of capabilities and revision rules that an agent maintains is the same throughout time, we can define the concept of *agent configuration* which is used to represent (the variable part of) the *state of an agent* at a given time. We simplify the definition given in [9] to the propositional version of 3APL.

Definition 3 (Agent Configuration). *[9] An agent configuration is represented by the tuple $\langle \sigma, \gamma, \Pi \rangle$, where σ is the agent's Belief Base. γ is the agent's Goal Base, such that for any ϕ such that $\gamma \models \phi$, we have that $\sigma \not\models \phi$. $\Pi \subseteq \mathcal{L}_P \times \mathcal{L}_G$ is the plan base of the agent.*

The semantics of the belief and goal query formulas entailment in the propositional 3APL is quite straightforward and will not be *explicitly* defined. The reader is invited to read [9] for further information.

[3] We differ from the notation used in [9], where the conjunction symbol $\wedge$ is used to represent the conjunction of goals.

As mentioned earlier, the agent uses the mental actions to update its beliefs. The update of the belief base of the 3APL agent is done in a quite simple way, by *removing* or *adding* facts to the belief base. Informally, when the precondition β, of a mental action $\langle \beta, \alpha, \beta' \rangle$, is believed by the agent, it will add, as a fact in its belief base, the literals in the postcondition β' that are not negated and remove the ones that are negated. The formal definition can be found in [9].

After performing a communication action $Send(r, type, A)$, a fact, *sent(r,type, A)*, stating that a message A, of type *type*, was sent to the agent r, is included in the belief base of the sending agent. A similar fact, *received(s,type,A)*, is included in the receiving agent's belief base, stating that a message A of type *type* was send by the agent *s*. Notice that messages exchanged between agents are only positive atoms, as no rules can be communicated.

An agent in 3APL uses its *Reasoning Rules* to adopt or change plans. There are two types of Reasoning Rules: the *Goal Planning Rules* and the *Plan Revision Rules*, the former being used by the agent to pursue a new goal and build a initial plan, and the later being used to revise a previously existing plan to obtain another plan. It maybe possible that one or more rules are applicable in a certain agent configuration, and the agent must decide which one to apply. In 3APL this decision is made through a *deliberation cycle*. Further details about the deliberation cycle can also be found in [9]. Here, we will not deal with the 3APL reasoning rules.

3 Modified Syntax

In this Section, we are going to begin to address the 3APL's limitations that we discussed previously, namely its limited capacity of updating an agent's beliefs and its limited expressive power of negative information. We introduce in the following definitions the syntax of the modified 3APL that we propose.

We begin modifying the agent configuration, by replacing the old belief base (σ) by a DLP. The goal base (γ) and the plan base (Π) are as in the original agent configuration.

Definition 4 (Modified Agent Configuration). *The Modified Agent Configuration is the tuple $\langle \sigma, \gamma, \Pi \rangle$, where σ is a DLP representing the agent's belief base. γ, Π are as before, representing, respectively, the agent's goal base and plan base.*

Now we modify the 3APL belief query language, by incorporating two types of negation, *negation by default* and *strong negation*. This will make it possible for the agent to reason with the open and closed world assumptions as we will investigate in the next Section.

Definition 5 (Modified Belief Query Language). *Let $\phi \in \mathcal{L}^{\neg, not}$. The modified belief query language, $\mathcal{L}_{\mathcal{B}}^{\mathcal{M}}$ is defined as follows:*

- $\top \in \mathcal{L}_{\mathcal{B}}^{\mathcal{M}}$;
- $B(\phi) \in \mathcal{L}_{\mathcal{B}}^{\mathcal{M}}$;
- $\beta_M, \beta'_M \in \mathcal{L}_{\mathcal{B}}^{\mathcal{M}}$ *then* $\beta_M \wedge \beta'_M \in \mathcal{L}_{\mathcal{B}}^{\mathcal{M}}$;
- $\beta_M, \beta'_M \in \mathcal{L}_{\mathcal{B}}^{\mathcal{M}}$ *then* $\beta_M \vee \beta'_M \in \mathcal{L}_{\mathcal{B}}^{\mathcal{M}}$.

Notice that differently from the Belief Query formulas in 3APL, the modified queries don't include symbols like $\neg B(\phi)$. As we will discuss with more details in the next Section, we don't feel the need for these type of symbols since the belief operator, $B(.)$, can have a literal, ϕ, as a parameter and not only an atom.

Now that we are considering the belief base of the agent as a Dynamic Logic Program, we will be able to update the belief base with a Generalized Logic Program. As in 3APL, the agent uses mental actions to update its belief base, but we will now consider the postcondition of these actions to be a Generalized Logic Program.

Definition 6 (Modified Mental Actions Specifications). *Let α_M be a modified mental action name, $\beta_M \in \mathcal{L}_{\mathcal{B}}^{\mathcal{M}}$ be the precondition of the action and P a GLP. Then, a modified mental action is a tuple $\langle \beta, \alpha_M, P \rangle$, and $ModAct$ is the set of all modified mental actions.*

$\langle B(tv\_on), turn\_off, \{not\, tv\_on \leftarrow\} \rangle$ is an example of a modified mental action representing the action of turning off the tv. Throughout this paper, we will explore the possibilities of using theses type of actions and give many other examples of application.

In a similar way, we modify the syntax of the communication actions by considering that the message in these actions are GLPs.

Definition 7 (Modified Communication Actions Specifications). *Let s be an agent name, type a performative or speech act and P a GLP. Then, a modified communication action is defined as $Send\,(s, type, P)$, and $ComAct$ as the set of all modified communication actions.*

$Send\,(user, inform, \{not\, power\_failure \leftarrow\})$ is an example of the modified communication action informing the *user* agent that the *power failure* ended.

4 Modified Semantics

In this Section, we define the semantics of the modified system, beginning with the semantics of the *belief query* formulas and afterwards of the *modified mental* and *communication* actions.

4.1 Modified Belief Query Semantics

The semantics of the Belief Queries will depend on the type of approach the agents adopt to handle the multiple stable models of a DLP. As we discussed previously, we consider three approaches: *Skeptical* ($\models_\cap$), *Credulous* ($\models_\cup$) and *Casuistic* ($\models_\Omega$). The consequences of choosing anyone of theses approaches are

not completely clear. More investigation will be needed to determine exactly in what conditions would be more suitable to select one of them, and therefore, we leave the belief query semantics conditioned to the approach used to determine the valuation of the agent's belief base.

Definition 8 (Semantics of Modified Belief Queries). *Let* $B(\phi), \beta_M, \beta'_M \in \mathcal{L}_{\mathcal{B}}^{\mathcal{M}}$ *be belief query formulas,* $\langle \sigma, \gamma, \Pi \rangle$ *be a modified agent configuration and* $x \in \{\cap, \cup, \Omega\}$ *. Then, the semantics of belief query formulas,* $\models_B$*, is defined as follows:*

$$\langle \sigma, \gamma, \Pi \rangle \models_B \top$$
$$\langle \sigma, \gamma, \Pi \rangle \models_B B(\phi) \Leftrightarrow \sigma \models_x \phi$$
$$\langle \sigma, \gamma, \Pi \rangle \models_B \beta_M \wedge \beta'_M \Leftrightarrow \langle \sigma, \gamma, \Pi, \Omega \rangle \models_B \beta_M \text{ and } \langle \sigma, \gamma, \Pi, \Omega \rangle \models_B \beta'_M$$
$$\langle \sigma, \gamma, \Pi \rangle \models_B \beta_M \vee \beta'_M \Leftrightarrow \langle \sigma, \gamma, \Pi, \Omega \rangle \models_B \beta_M \text{ or } \langle \sigma, \gamma, \Pi, \Omega \rangle \models_B \beta'_M$$

We don't feel the need to include in the belief query language the negation of belief literals,$\neg B\,(\phi)$, since with the definition above, the programmer has the possibility of using the open the closed world assumptions by using query formulas of the type $B\,(\neg\phi)$ and $B\,(not\,\phi)$, respectively. Consider the following illustrative example:

Example 2. Let the belief base of an agent consist of the following facts:

$$\{p(a) \leftarrow \qquad\qquad p(b) \leftarrow\}$$

If in the original 3APL, we propose the query $\neg B\,(p\,(c))$ it will succeed, since it is not possible to *unify* $p(c)$ with any of the given facts, and the negation by finite failure will succeed. Hence, the 3APL agents use the closed world assumption.

This query could be done in a similar way in the modified 3APL, by using the modified belief query $B\,(not\,p\,(c))$. As the program above has an unique stable model, namely $\{p(a), p(b)\}$, it would represent the beliefs of the agent. Reminding the definition of the entailment of the default negation: if $\phi \notin M$ then $M \models not\,\phi$. The modified belief query will also succeed, since $p(c) \notin \{p(a), p(b)\}$.

4.2 Semantics of Action Execution

In this subsection we formalize the semantics of the actions in this modification of 3APL.

We start with a definition that formalizes the semantics of the Modified Mental Actions. Informally, if the precondition (β_M) of the modified mental action (α_M) is *satisfied* by the agent configuration, the belief base of the agent will be *updated* with the program (P) in the postcondition of the action. Syntactically, this update adds a new program at the end of the sequence of programs, that composes the Belief Base. We then use the semantics of Dynamic Logic Programming to characterize this updates.

Definition 9 (Semantics of Modified Mental Actions). *Let* $\langle \beta_M, \alpha_M, P \rangle$*,* $\langle \sigma, \gamma, \Pi \rangle$*, be, respectively, a modified mental action and modified agent*

configuration, $x \in \{\cap, \cup, \Omega\}$*, and* $\kappa \in \mathcal{L}_G$ *The semantics of the modified mental action is given by the transition rule:*

$$\frac{\langle \sigma, \gamma, \{(\alpha_M, \kappa)\}\rangle \models_B \beta_M}{\langle \sigma, \gamma, \{(\alpha_M, \kappa)\}\rangle \rightarrow \langle (\sigma, P), \gamma', \{(\epsilon, \kappa)\}\rangle}$$

where $\gamma' = \gamma \setminus \{\Sigma \mid \Sigma \subseteq \mathcal{K} \wedge (\sigma, P) \models_x \Sigma\}$.

This modification in the definition of Mental Action greatly increases the *expressiveness* of the language. Now the agent can use generalized logic programs instead of simple facts to update the belief base. Furthermore, the semantics of DLPs gives us an *intuitive solution* for the conflicting cases, by automatically rejecting older rules if they are *conflicting* with a newer ones.

For example, consider again the situation explained in the Introduction, where the agent has a belief base consisting of the program:

$$mother\_said(santa\_claus) \leftarrow$$
$$believe(santa\_claus) \leftarrow mother\_said(santa\_claus)$$

And after a mental action it would have to conclude that $believe(santa\_claus)$ is no longer true. This can be easily done by updating the belief base with the program $\{not\ believe(santa\_claus) \leftarrow\}$. Then, the DLP semantics will reject the rule $believe(santa\_claus) \leftarrow mother\_said(santa\_claus)$ and the agent will no longer believe in $believe(santa\_claus)$ but still believe in $mother\_said(santa\_claus)$.

Even though we believe that the semantics of DLP can handle most of the conflicting cases in an elegant manner, there are some cases that require *program revision*. Note that revision and updates are two different forms of belief change [13]. To achieve both forms of belief change, would be necessary to include a mechanism that would make it possible for the programmer to customize the revision of the programs, for example, by programming the deliberation cycle. We will not approach this issue in this paper.

The final modification that we propose for the 3APL actions concerns the communication actions. In 3APL the agent uses communication actions to send messages to other agents in the system. Up to now the messages that the agents transmit are positive facts. Since our agents have the possibility to update their beliefs with GLPs, it makes sense to use this added expressiveness and allow GLPs to be exchanged between the agents. Accordingly, in this proposal, the agents will exchange messages containing Generalized Logic Programs.

In a similar way as done in 3APL, after performing a communication action $(Send(r, type, P))$, the sending agent (s) will update its belief base with the program $\{sent(r, type, P) \leftarrow\}$ and the receiving agent (r) updates its belief base with the program $\{received(r, type, P) \leftarrow\}$[4].

By combining modified communication and mental actions, agents are now able to update their belief base with knowledge that they receive. Normally, an

[4] Programs can be associated with identifiers to be used when the facts sent(.) or received(.) are added in the belief base to represent these programs.

agent has a *social point of view* about the other agents in the environment, and may consider the information passed by another *trustworthy* agent to be true. For example, it is usually the case that a son believes what his father tells him. This could be represented using the following modified mental action:

$$\langle \{B(received(father, command, P)) \wedge B(obey(father))\}, obey\_father, \{P\} \rangle$$

where the agent would update its belief base with the program P, if it believes that it should obey his father and that it received from his father a command containing the message P.

5 Properties of the Modified 3APL

In this Section, we elaborate on the features provided by the modification of the 3APL system proposed in this paper.

Evolving Knowledge Bases - By adopting their belief bases as Dynamic Logic Programs and using its semantics to solve the possible conflicts when updating its beliefs, 3APL agents can have *evolving belief bases*. This dynamic character of its knowledge base opens the possibility of performing more complex updates using generalized logic programs instead of adding or removing facts. Agents with this modification can learn new rules even though they *partially* conflict with previous knowledge. For example, an agent may consider that all the published papers are good, represented by the GLP $\{papers\_good(X) \leftarrow\}$. Then, it learns that not all papers are good because the ones published in poor venues are not so good, hence updates its beliefs with the program $\{not\ papers\_good(X) \leftarrow bad\_congress(X)\}$. Notice that if the agent *doesn't believe* the paper X is from a *bad congress* it will use the previous knowledge and consider the *paper as good*. However if it *believes* that the paper X comes from a *bad congress* the newer rule will reject the older one. More about evolving knowledge bases can be found in [14];

The next proposition states that in fact, all DLPs can be semantically represented by an agent in the modified 3APL.

Proposition 1. *Let $\mathcal{P}$ be a DLP, $x \in \{\cap, \cup, \Omega\}$ and $\langle \mathcal{P}, \gamma, \Pi \rangle$ be a modified agent configuration. Then:*

$$(\forall L \in \mathcal{L}^{\neg, not}).(\mathcal{P} \models_x L \Leftrightarrow \langle \mathcal{P}, \gamma, \Pi \rangle \models_x B(L))$$

Proof: Trivial from the definition of the modified belief queries.

Strong and Default Negation - Agents in 3APL treat negation as *negation by failure*. In the modification proposed in this paper, we increase considerably the expressiveness of the agents by introducing strong as well as the default negation. This allows the agents to reason with a *closed or open world assumption*.

Consider the *classical car - train cross* example, where the car wants to cross the rails but it must be *sure* that a train is not coming. We can use the following two modified mental actions to model this situation:

$$\langle \{B(\neg train)\}, cross, \{crossed \leftarrow\}\rangle$$
$$\langle \{B(not\, train) \wedge B(not\, \neg train)\}, listen, \{\neg train \leftarrow \neg sound\}\rangle$$

The first action is of passing the cross when the agent is sure that there is no train coming ($\neg train$). While the second action illustrates the use of the default negation to represent doubt, since the agent will listen when it doesn't know for sure if the train is coming ($not\, train$) or not coming ($not\, \neg train$). This situation was not possible to be modeled in the original 3APL.

From [15], we know that Dynamic Logic Programming is a generalization of Answer Set Programming. Together with the proposition 1, we obtain the following corollary stating that in fact, the agent belief semantics in the modified architecture also generalizes Answer Set Programming.

Corollary 1. *Let $\mathcal{P}$ be an ASP, $x \in \{\cap, \cup, \Omega\}$, and $\langle(\mathcal{P}), \gamma, \Pi\rangle$ be a modified agent configuration. Then:*

$$(\forall L \in \mathcal{L}^{\neg, not}).(\mathcal{P} \models_x L \Leftrightarrow \langle(\mathcal{P}), \gamma, \Pi\rangle \models_x B(L))$$

More Expressive Communications - Agents in 3APL communicate through messages containing only facts. By proposing agents that can communicate programs to other agents, we increasing the possibilities of the multi-agent system. Agents can share knowledge represented by rules. Furthermore, depending on the semantics of the exchanged programs, they could also represent plans or explanations about the environment [4]. As discussed in the previous sections, the agents could update their belief base with theses programs;

Nondeterministic Effect of Actions - As discussed in [3], we can use the multiple stable models of a Generalized Logic Program to represent nondeterministic effects of mental actions. Consider the mental action representing the action of shooting in the famous Yale shooting problem, where the agent tries to kill a turkey with a shot gun, but after shooting, it can happen that the agent misses the turkey:

$$\langle B\,(shoot)\,, shoot, \{kill\_turkey \leftarrow not\, miss;\, miss \leftarrow not\, kill\_turkey\}\rangle\,;$$

There are two possible effects for the action shoot, one if the agent shot the turkey and therefore killed it and another where the agent missed and the turkey is presumably alive.

NP-Complete Complexity - To have the increase in the expressiveness of the language, as investigated in the points above, there is an increase in the complexity of the agent. According to [15] the complexity of computing the stable models is *NP-Complete*.

6 Example

In this Section, we give an example that could be straightforwardly implemented in our modified 3APL system.

Consider the scenario, where *007* is in one of his mission for the *MI6*, to save the world. After infiltrating the enemy base, our special agent encounters the control room where it is possible to deactivate the missile that is threatening to destroy the world as we know it. However, since he was meeting one of the bond girls for dinner, he didn't attend the classes of *Mr. Q* on how to deactivate bombs.

We can represent his belief base as follows:

$$\{\, save\_world \leftarrow \neg bomb \,\}$$

At this point the agent is not able to save the world, since the program has one stable model, namely $\emptyset$. But our agent remembers the briefing of *Mr. Q* before this mission, when *Mr. Q* explained about a special device installed in his watch that could be used to contact the *MI6* headquarters. He immediately takes a look at his watch, presses the special button installed, and asks for further instructions, represented by the communication action, $Send(MI6, request, \{help \leftarrow\})$. The *MI6* headquarters, unable to find *Mr. Q*, sends him some instructions that could be an incorrect one, represented by the following program, P_{MI6}:

$$P_{MI6} : \left\{ \begin{array}{ll} know\_deactivate & \leftarrow not\, wrong\_instructions \\ wrong\_instructions & \leftarrow not\, know\_deactivate \end{array} \right\}$$

Since *007* trusts *MI6*, he updates its beliefs with the modified mental action:

$$\langle B\left(received\left(MI6, inform, P_{MI6}\right)\right), listen, P_{MI6}\rangle\,;$$

With this update, the agent's belief base supports two stable models:

$$\{wrong\_instructions, received(MI6, inform, P_{MI6}\} \text{ and}$$
$$\{know\_deactivate, \neg bomb, save\_world, received(MI6, inform, P_{MI6}\}$$

Notice that the agent must handle the multiple stable models. We consider that for the task of *saving the world* a more conservative approach should be used, namely a *Skeptical* one (where the intersection of all the models is used to determine the agent's beliefs).

Now the spy has to acquire more information about the bomb, since he is not sure if it is possible to deactivate the bomb with the instructions given. If he tries to disable the bomb with the acquired information there can be two outcomes, that the bomb is disabled or that the missile is launched. Represented by the following modified mental action:

$$\langle B\left(not\, know\_deactivate\right), disable\_with\_risk, P_{disable}\rangle$$

where:

$$P_{disable} : \left\{ \begin{array}{ll} \neg bomb & \leftarrow not\, missile\_launched \\ missile\_launched & \leftarrow not\, \neg bomb \end{array} \right\}$$

Therefore, he takes a look at the room (sensing action)[5], and finds the manual of the bomb and realizes that the instructions given were not wrong, updating once more his beliefs with the program:

$$\{\, not\, wrong\_instructions \leftarrow \}$$

With this new knowledge the spy is able to conclude that he knows how to deactivate the bomb (*know_deactivate*), and therefore he is able to disable the bomb ($\neg bomb$), using the following modified mental action:

$$\langle B\,(know\_deactivate)\,, disable\_without\_risk, \{\neg bomb \leftarrow\}\rangle$$

After this action, *007* has safely deactivated the bomb ($\neg bomb$) and finally saved the world (*save_world*) once more (to follow precisely the *007* movies it would be necessary to include somewhere at the end a bond girl...).

In this example we were able to demonstrate several aspects that can be used in the modified 3APL proposed here. First, the use of the strong negation ($\neg bomb$), since it could be incorrect to conclude that the spy saved the world if we used instead default negation (*not bomb*), because there would still be a chance that the bomb is activated but the agent doesn't know it. Second, it was possible to send rules in the communication actions (when the *MI6* headquarters sends 007 the instructions) instead of simple facts. Third, if the agent tried to disable the bomb without the assurance that the information given is correct, there would be a nondeterministic effect after performing the *disable_with_risk* action (bomb being disabled or launching the missile). Finally, we could demonstrate the knowledge evolution, when the agent senses that the instructions were right (*not wrong_instructions* $\leftarrow$), the previous rule (*wrong_instructions* $\leftarrow$ *not know_deactivate*) is rejected and it is finally possible for the agent to save the world (*save_world*).

7 Conclusions

In this paper we proposed a modification to the syntax and semantics of the 3APL language. We investigated the main properties that are obtained by having an agent with a belief base represented by Dynamic Logic Program. The modification proposed considerably increases the expressiveness of the language, by allowing *knowledge updates*, *strong* and *default* negation, more *expressive communication* between the agents. However, to be able to have this expressiveness, there is a clear increase in the complexity of the system.

We investigate in [21], the properties obtained by representing the agent's goal base by a DLP. The agent programmer can elegantly *adopt*, *drop* goals, as well as represent *achievement* and *maintenance* goals. We believe that there would

[5] Notice that we did not deal in this paper with sensing actions, i.e., external actions in the 3APL. However, as the environment is considered as a set of facts, these type of action can be straightforwardly incorporated in our system by updating the agent's beliefs with the sensing information.

be much synergy, if the approaches used here and the approaches in [21] were joined in an unique agent framework.

Even though we believe that the semantics of DLP can handle most of the conflicting cases in an elegant manner, there are some cases that require *program revision*. It would necessary to include a mechanism that would make it possible for the programmer to customize the revision of the programs, for example, by programming the deliberation cycle.

[17] presents a way to represent the social point of view of agents using *Multi Dimensional Dynamic Logic Programs* (MDLP). Further research could be made to try to incorporate these social point views in the 3APL agents, and use this view to decide to consider information sent by another agent or to decide the goals of an agent. A mechanism to update the MDLP would have to be defined, possibly in a similar line as KABUL [14] or MLUPS [16].

References

1. J. J. Alferes, F. Banti, A. Brogi, and J. A. Leite. The refined extension principle for semantics of dynamic logic programming. *Studia Logica*, 79(1):7–32, 2005.
2. J. J. Alferes, J. Leite, L. M. Pereira, H. Przymusinska, and T. Przymusinski. Dynamic updates of non-monotonic knowledge bases. *Journal of Logic Programming*, 45(1-3):43–70, 2000.
3. C. Baral. Reasoning about actions: Non-deterministic effects, constraints, and qualification. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montral, Qubec, Canada, August 20-25 1995*, volume 2, pages 2017–2026. Morgan Kaufmann, 1995.
4. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.
5. R. Bordini, J. Hübner, and R. Vieira. Jason and the Golden Fleece of agent-oriented programming. In Bordini et al. [6], chapter 1.
6. R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors. *Multi-Agent Programming: Languages, Platforms and Applications*. Number 15 in Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer, 2005.
7. R.H. Bordini, L. Braubach, M. Dastani, A. El F. Seghrouchni, J.J. Gomez-Sanz, J. Leite, G. O'Hare, A. Pokahr, and A. Ricci. A survey of programming languages and platforms for multi-agent systems. *Informatica*, 30(1):33–44, 2006.
8. S. Constantini and A. Tocchio. A logic programming language for multi-agent systems. In S. Flesca, S. Greco, N. Leone, and G. Ianni, editors, *Logics in Artificial Intelligence, European Conference, JELIA 2002, Cosenza, Italy, September, 23-26, Proceedings*, volume 2424 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2002.
9. M. Dastani, M. B. van Riemsdijk, and J.-J. Ch. Meyer. Programming multi-agent systems in 3APL. In *Multi-Agent Programming: Languages, Platforms and Applications*, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, chapter 2. Springer, 2005.
10. J. Dix and Y. Zhang. IMPACT: a multi-agent framework with declarative semantics. In Bordini et al. [6], chapter 3.
11. M. Gelfond and V. Lifschitz. Logic programs with classical negation. In Warren and Szeredi, editors, *7th International Conference on Logic Programming*, pages 579–597. MIT Press, 1990.

12. 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.
13. H. Katsuno and A. O. Mendelzon. On the difference between updating a knowledge base and revising it. In J. A. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91).*, pages 387–394. Morgan Kaufmann, 1991.
14. J. Leite. *Evolving Knowledge Bases*. IOS press, 2003.
15. J. Leite. On some differences between semantics of logic program updates. In C. Lemaître, C. A. Reyes, and J. A. González, editors, *Advances in Artificial Intelligence - IBERAMIA 2004, 9th Ibero-American Conference on AI, Puebla, México, November 22-26, 2004, Proceedings*, volume 3315 of *Lecture Notes in Computer Science*, pages 375–385. Springer, 2004.
16. J. Leite, J. J. A., L. M. Pereira, H. Przymusinska, and T. Przymusinski. A language for multi-dimensional updates. In J. Dix, J. A. Leite, and K. Satoh, editors, *Computational Logic in Multi-Agent Systems: 3rd International Workshop, CLIMA'02, Copenhagen, Denmark, August 1, 2002, Pre-Proceedings*, volume 93 of *Datalogiske Skrifter*, pages 19–34. Roskilde University, 2002.
17. J. Leite, J. J. Alferes, and L. M. Pereira. On the use of multi-dimensional dynamic logic programming to represent societal agents' viewpoints. In P. Brazdil and A. Jorge, editors, *Progress in Artificial Intelligence, Knowledge Extraction, Multi-agent Systems, Logic Programming and Constraint Solving, 10th Portuguese Conference on Artificial Intelligence, EPIA 2001, Porto, Portugal, December 17-20, 2001, Proceedings*, volume 2258 of *Lecture Notes in Computer Science*, pages 276–289. Springer, 2001.
18. J. Leite, J. J. Alferes, and L. M. Pereira. Minerva - a dynamic logic programming agent architecture. In *Intelligent Agents VIII*, volume 2333 of *LNAI*. Springer, 2002.
19. J. Leite and L. M. Pereira. Generalizing updates: From models to programs. In J. Dix, L. M. Pereira, and T. C. Przymusinski, editors, *Logic Programming and Knowledge Representation, Third International Workshop, LPKR '97, Port Jefferson, New York, USA, October 17, 1997, Selected Papers*, volume 1471 of *Lecture Notes in Computer Science*, pages 224–246. Springer, 1998.
20. V. Mascardi, M. Martelli, and L. Sterling. Logic-based specification languages for intelligent software agents. *Theory and Practice of Logic Programming*, 4(4), 2004.
21. V. Nigam and J. Leite. Using dynamic logic programming to obtain agents with declarative goals. In M. Baldoni and U. Endriss, editors, *Pre-Procs. of the 4th International Workshop on Declarative Agent Languages and Technologies, (DALT06), Hakodate, Japan, 2006*, 2006.
22. M. Thielscher. *Reasoning Robots: The Art and Science of Programming Robotic Agents*. Springer, 2005.