Using Dynamic Logic Programming to Obtain Agents with Declarative Goals – preliminary report

Vivek Nigam[⋆] and João Leite

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

Abstract. Goals are used to define the behavior of (pro-active) agents. It is our view that the goals of an agent can be seen as a knowledge base of the situations that it wants to achieve. It is therefore in a natural way that we use Dynamic Logic Programming (DLP), an extension of Answer-Set Programming that allows for the representation of knowledge that changes with time, to represent the goals of the agent and their evolution, in a simple, declarative, fashion. In this paper, we represent agent's goals as a DLP, discuss and show how to represent some situations where the agent should adopt or drop goals, and investigate some properties that are obtained by using such representation.

1 Introduction

It is widely accepted that *intelligent agents* must have some form of *pro-active* behavior [19]. This means that an intelligent agent will try to pursue some set of states, represented by its *goals*. At the same time, goals will serve as explanations for agent's *actions*. Goals have two distinct, though related, aspects: a *procedural* that can be identified with the sequence of actions that the agent attempts to perform in order to achieve a goal; and a *declarative* that can be associated with the set of states that the agent wants to bring about. In this paper we will focus on the declarative aspect of goals.

Recently, there has been an increasing amount of research devoted to the issue of declarative goals and their properties [18, 6, 13, 16, 17, 15]. Agent programming languages with declarative goals open up a number of interesting possibilities to the programmer, such as checking if a goal has been achieved, if a goal is impossible, if a goal should be dropped, i.e., if the agent should stop pursuing a goal, or if there is interference between goals [18, 15]. Having declarative goals also facilitates the task of constructing agents that are able to *communicate* them with other agents [13]. In [18, 15, 13] the reader can find examples illustrating the need for a declarative aspect to goals.

As dynamic entities, agents often must adopt new goals, and drop existing ones, and these changes in the adopted goals can be made dependent on the

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

state of affairs. There has been a great deal of research to identify when an agent should change its goals [5, 16, 15, 18]. For example, an agent should drop a goal when it believes that the goal is no longer achievable (maybe represented by a *failure condition* [18]). As for adopting new goals, after a negotiation is successfully closed there may be new *obligations* [7] that the agents involved have committed to, that lead to the revision of the agent's goals and, possibly, the adoption of new ones.

In this paper, we will address the problem of representing and reasoning about dynamic declarative goals using a logic programming based approach.

In [12, 8], the paradigm of *Dynamic Logic Programming (DLP)* was introduced. According to *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 they are kept for as long as they are not in conflict with newly added ones, these always prevailing.

It is our perspective that the declarative goals of an agent can be seen as a knowledge base encoding the situations it wants to achieve. There has been, in the past years, an intense study of the properties of DLP to represent knowledge bases that evolve with time [2, 8, 11]. However, up to now, there hasn't been much investigation of how DLP could be used to represent, in a declarative manner, the goals of an agent. Since DLP allows for the specification of knowledge bases that undergo change, and enjoys the expressiveness provided by both strong and default negations, by dint of its foundation in answer-set programming, it seems a natural candidate to be used to represent and to reason about the declarative goals of an agent, and the way they change with time.

In this paper we will represent the *goal base* of an agent as a *Dynamic Logic Program*, and investigate some of its properties. Namely, we will see that the semantics of DLP will allow us to straightforwardly *drop* and *adopt* new goals by updating the goal base of the agent, and will allow those operations to be conditional on the current state of affairs.

Furthermore, an agent can distinguish between *maintenance and achievement* goals. A maintenance goal represents a state of affairs that the agent wants to hold in all states. For example, a person doesn't want to get hurt. An achievement goal represents a state of affairs that, once *achieved*, is no longer pursued. For example, an agent that has as goal to write a paper for a congress, after it believes it has written the paper, it should no longer consider this as a goal. Therefore,

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

to correctly define the conditions for dropping a goal, we also investigate how to express maintenance and achievement goals using DLP.

For our purpose, we will use a simple agent framework to be able to clearly demonstrate the properties obtained by using DLP. Agents in this framework are composed of data structures representing its beliefs, goals, committed goals (intentions) and reasoning rules. We propose three types of reasoning rules: **1)** Intention Adoption Rule: used to *commit* to a goal by adopting a plan to achieve it; **2)** Goal Update Rule: used to *update* an agent's goals using the DLP semantics; **3)** Intention Dropping Rule: used to *drop* previously committed goals.

The remainder of the paper is structured as follows: in the next Section we are going to present some preliminaries, introducing DLP and the agent framework we are going to use. Later, in Section 3, we are going to define the semantics of the goal queries and in Section 4 the reasoning rules. In Section 5, we discuss some situations related to when to drop and adopt new goals, and how to use the DLP semantics to represent these situations. In Section 6 we give a *simple* example of a multi-agent system illustrating how DLP could be used to represent goals, to finally draw some conclusions and propose some further research topics in Section 7.

2 Preliminaries

In this section we are going to give some preliminary definitions that will be used throughout the paper. We start by introducing the language and semantics of *Dynamic Logic Programming* and, afterwards, we introduce the simple *agent framework* that we will adopt to demonstrate our investigations.

2.1 Dynamic Logic Programming

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. The set of objective literals is denoted by $\mathcal{L}_{\mathcal{K}}^{\neg}$ and the set of literals by $\mathcal{L}_{\mathcal{K}}^{\neg,not}$. 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*

[2] Expanded programs are defined to appropriately deal with strong negation in updates. For more on this issue, the reader is invited to read [9, 8]. From now on, and

M of $\mathcal{K}$ is a set of objective literals that is consistent i.e., M does not contain both A and $\neg A$. We define the set $\mathcal{I}$ as the set of all interpretations. 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. Only an inconsistent set of objective literals, In, will entail the special symbol $\bot$ (denoted by $In \models \bot$). $\bot$ can be seen semantically equivalent to the formula $A \wedge \neg A$. 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 $\mathcal{K}$. As usual, we will consider all the variables appearing in the programs as a shorthand for the set of all their possible ground instantiations.

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 $\mathcal{P} \cup \mathcal{P}''$ to denote $(P_1 \cup P_1'', ..., P_s \cup P_s'')$.

In the past years there have appeared several semantics for a DLP. We are going to use the *Refined Dynamic Stable Model* semantics defined below, because of its nice properties, as investigated in [9].

Definition 1 (Semantics of DLP). *[8, 1] Let $\mathcal{P} = (P_1, \ldots, P_s)$ be a dynamic logic program over language $\mathcal{K}$, A be an objective literal, $\rho\,(\mathcal{P})$, M' and $least(.)$ be as before. An interpretation M is a (refined dynamic) stable model of $\mathcal{P}$ iff*

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

Where:

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

It is important to notice that a DLP might have more than one stable model. Each stable model can be viewed as a *possible world* that follows from the knowledge represented by the DLP. We will denote by $SM(\mathcal{P})$ the set of stable models of the DLP $\mathcal{P}$. Further details and motivations concerning DLPs and its semantics can be found in [8].

2.2 Agent Framework

In this subsection we are going to define the agent framework[3] that we will use throughout this article. We will start by introducing the concept of *agent*

unless otherwise stated, we will always consider generalized logic programs to be in their expanded versions.

[3] The agent framework defined in this section could be seen as a modified (simplified) version of the agent framework used in the 3APL multi-agent system [4].

configuration, which consists of a *belief base* representing what the agent believes the world is, a *goal base* representing the states the agent wants to achieve, a set of *reasoning rules* and a set of *intentions* with associated *plans* representing the goals that the agent is currently *committed* to achieve. We are going to make precise, later in Section 4, how the reasoning rules of the agents are defined. We are considering that the agent has, at its disposal, a *plan library* represented by the set of plans, *Plan*. A plan can be viewed as a *sequence of actions* that can modify the agent's beliefs or/and the environment surrounding it, and is used by the agent to *try* to achieve a committed goal.

Our main focus in this paper is to investigate the properties of representing the goal base as a Dynamic Logic Program. We are not going to give the *deserved* attention to the belief base. We consider the belief base as a simple interpretation. However, a more complex belief base could be used. For example, we could represent the belief base also as a Dynamic Logic Program and have some mechanism such that the agent has an unique model for its beliefs[4]. Elsewhere, in [14], we explore the representation of 3APL agent's belief base as a DLP.

Definition 2 (Agent Configuration). *An agent configuration is a tuple* $\langle \sigma, \gamma, \Pi, R \rangle$*, where* $\sigma \in \mathcal{I}$ *is an interpretation representing the agent's belief base,* γ *a Dynamic Logic Program representing it's goal base,* $\Pi \subseteq Plan \times \mathcal{L}^{\neg}$ *the intentions of the agent and* R *the set of reasoning rules.*

We assume that the semantics of the agents is defined by a *transition system*. A transition system is composed of a set of *transition rules* that transforms one agent configuration into another agent configuration, in one computation step. It may be possible that one or more transition rules are applicable in a certain agent configuration. In this case, the agent must decide which one to apply. This decision can be made through a *deliberation cycle*, for example, through a priority among the rules. In this paper, we won't specify a deliberation cycle. An unsatisfied reader can consider a *non-deterministic* selection of the rules.

We are interested in knowing what an agent believes and what are its goals. To this purpose, we start by introducing, in the next definition, the *belief and goal query languages.*

Definition 3 (Belief and Goal Query Language). *Let* $\phi \in \mathcal{L}^{\neg,not}$ *and* $\phi' \in \mathcal{L}^{\neg}$*. The belief query language,* $\mathcal{L}_{\mathcal{B}}$*, with typical element* β*, and the goal query language,* $\mathcal{L}_{\mathcal{G}}$*, with typical element* κ *are defined as follows:*

$$\begin{array}{lll} \top \in \mathcal{L}_{\mathcal{B}} & B\phi \in \mathcal{L}_{\mathcal{B}} & \beta, \beta' \in \mathcal{L}_{\mathcal{B}} \text{ then } \beta \wedge \beta' \in \mathcal{L}_{\mathcal{B}} \\ \top \in \mathcal{L}_{\mathcal{G}} & G\phi' \in \mathcal{L}_{\mathcal{G}} & \end{array}$$

Notice that we don't include default literals in the goal query formulas. This is because we believe that it would only make sense for an agent to pursue a situation that the agent is completely sure when it is achieved. For example, if

[4] For example, a *belief model selector* that would select one of the stable models of the belief base to represent the agent's beliefs.

an agent had the goal of not (by default) failing an exam, $not\ fail$, it would be possible for the agent not to study for the exam and still satisfy this goal (considering that the agent is not a *genius*) by simply not checking its mark. On the other hand, default literals can be quite useful for the belief queries. For example, for cautious agents in emergency situations, if an agent is not sure that a place is safe ($not\ safe$), it could trigger the goal of moving to a safer location. We will explain better how this could be represented when we discuss goal adoption and dropping, in Section 5.

Now, we will start by defining the semantics of the belief query formulas ($\models_B$). The semantics of the goal query formulas ($\models_G$), one of the key interests of this paper, will be defined later in Section 3.

Definition 4 (Semantics of Belief Formulas). *Let $B\phi, \beta, \beta' \in \mathcal{L_B}$ be belief query formulas and $\langle \sigma, \gamma, \mathit{\Pi}, R \rangle$ be an agent configuration. Then, the semantics of belief query formulas, $\models_B$, is defined as:*

$$\langle \sigma, \gamma, \mathit{\Pi}, R \rangle \models_B \top$$
$$\langle \sigma, \gamma, \mathit{\Pi}, R \rangle \models_B B\phi \Leftrightarrow \sigma \models \phi$$
$$\langle \sigma, \gamma, \mathit{\Pi}, R \rangle \models_B \beta \wedge \beta' \Leftrightarrow \langle \sigma, \gamma, \mathit{\Pi}, R \rangle \models_B \beta \text{ and } \langle \sigma, \gamma, \mathit{\Pi}, R \rangle \models_B \beta'$$

Although this is a quite simple agent framework it will be enough for the purpose of this paper.

3 Semantics of Agent Goal Bases

As defined in the previous section, we are considering the goal base of the agent as a Dynamic Logic Program. We will use the stable models of the goal base of the agent to determine the goals that it should achieve. Since the logic programs used in DLP use default negation, we can have situations where one DLP has more than one stable model, each internally consistent, but entailing contradictory conclusions between them. For example, consider a goal base consisting of the logic program with the following two rules:

$$a \leftarrow not\, \neg a. \qquad \neg a \leftarrow not\, a.$$

This program has two stable models, namely $\{a\}$ and $\{\neg a\}$. Even though each of them is consistent (recall that models are interpretations which, themselves, are consistent), they are contradictory in the sense that one entails a while the other entails $\neg a$. This contradiction could be seen as undesirable. However, as argued by Hindriks et al. in [6], the goal base of an agent doesn't have to be consistent since, for example, the goals of an agent can be achieved at different times. We add to this that these apparently contradictory goals can just be seen as alternative ones. The semantics of the intention adoption rules, defined below, makes sure that the agent doesn't concurrently pursue inconsistent intentions[5].

[5] [17] uses a default logic system to be able to express contradictory goals, but no mechanism to drop goals is proposed. We propose a system based on the stable models of the goal base, with the same expressiveness as the system in [17], and with the possibility of elegantly drop goals.

However, we shouldn't directly consider the stable models of the goal base (γ) of an agent as its goals, because the agent shouldn't consider a goal if it already believes that the goal is currently achieved. A naive way of solving this problem is to refine the stable models of the goal base by removing the goals that are entailed by the belief base: $GM = \{M \setminus \sigma \mid M \in SM(\gamma)\}$. But, by doing so, we *partially* lose expressiveness of having *conditional goals*. Consider the following illustrative example:

Example 1 (Conditional Goals). Let the goal base of an agent be the DLP composed of one GLP, with the intended meaning that the agent has as goal to buy a Ferrari if it won the prize, otherwise it would like to buy a Beetle. The goal of getting an insurance will depend on which car the agent will buy.

$$\begin{aligned} buy\_ferrari &\leftarrow win\_lottery. \\ buy\_beetle &\leftarrow not\, win\_lottery \\ get\_insurance &\leftarrow buy\_ferrari. \end{aligned}$$

We must consider the agent's belief base to determine what its goals are. Which car to buy will depend on whether it believes to have won or not the lottery, since obviously winning the lottery would not be a feasible goal for the agent.

The next definition formalizes an agent's *Goal Models*. The agent's Goal Models will be used to represent the agent's goals and they are obtained by refining the stable models of the agent's goal base in such a way that the agent takes in consideration its beliefs, and doesn't consider a formula as a goal if this formula is entailed by its belief base. In the previous example, if $win\_lottery$ is entailed by the beliefs of the agent, $buy\_ferrari$ would be one of its goals.

Definition 5 (Goal Models). *Let $\langle \sigma, \gamma, \Pi, R \rangle$ be an agent configuration. Then, the set of Goal Models (GM) of the agent is defined as:*

$$GM\,(\sigma, \gamma) = \{M \setminus \sigma \mid M \in SM((\gamma, \Psi(\sigma)))\}$$

where $\Psi(\sigma) = \{L \leftarrow \mid L \in \sigma\}$

Notice that, similarly to interpretations, the Goal Models are individually consistent, but two different goal models can be mutually contradictory. As argued previously, we want to express agents with contradictory goals. Therefore, to express the goals of an agent, we are going to use simultaneously all of its Goal Models.

The definition below formalizes the semantics of the goal query formulas.

Definition 6 (Semantics of Goal Query Formulas). *Let $G\phi, \kappa, \kappa' \in \mathcal{L}_G$ be a goal query formula and $\langle \sigma, \gamma, \Pi, R \rangle$ be an agent configuration. Then, the semantics of goal query formulas, $\models_G$, is defined as:*

$$\begin{aligned} &\langle \sigma, \gamma, \Pi, R \rangle \models_G \top \\ &\langle \sigma, \gamma, \Pi, R \rangle \models_G G\phi \Leftrightarrow \exists M.(M \in GM(\sigma, \gamma) \wedge M \models \phi) \end{aligned}$$

The next proposition states that the agent cannot have a goal that is entailed by the belief base.

Proposition 1. *Let $\langle \sigma, \gamma, \Pi, R \rangle$ be an agent configuration, then:*

$$(\forall \phi \in \sigma).(\langle \sigma, \gamma, \Pi, R \rangle \not\models_G G\phi)$$

Proof. It is trivial from the way the Goal Models are constructed and by the Definition 6 of the Semantics of Goal Query Formulas

4 Reasoning Rules

We now define the types of reasoning rules an agent can have. We begin following [17], introducing the *Intention Adoption Rule* that is used by the agent to commit to a goal by associating a plan to it.

Definition 7 (Intention Adoption Rules). *Let $\beta \in \mathcal{L}_{\mathcal{B}}$ be a belief query formula and $\kappa \in \mathcal{L}_{\mathcal{G}}$ be a goal query formula, and $\pi \in Plan$ be a plan. The Intention Adoption Rules is defined as, $\kappa \leftarrow \beta \mid \pi$. We will call, β the guard of the rule and κ the head of the rule.*

Informally, the semantics of the Intention Adoption Rules is that if the goal base satisfies the head of the rule ($\kappa = G\phi$) and the agent beliefs in the guard (β) of the rule, the plan π is adopted to try to achieve the goal in the head of rule by adding the pair (π, ϕ) to the agent's intention base. However, as discussed by Bratman in [3], a *rational agent* shouldn't incorporate new intentions if it *conflicts* with the current intentions. For example, a rational agent wouldn't adopt the intention of going on vacations if it has committed to clean its house.

Taking this into account, we now formalize the semantics of the intention adoption rules.

Definition 8 (Semantics of Intention Adoption Rules). *Let $\langle \sigma, \gamma, \Pi, R \rangle$ be an agent configuration, $\kappa \leftarrow \beta \mid \pi \in R$, is an Intention Adoption Rule, where $\kappa = G\phi$, and $\Pi = \{(\pi_1, \phi_1), \ldots, (\pi_n, \phi_n)\}$.*

$$\frac{\langle \sigma, \gamma, \Pi, R \rangle \models_G \kappa \qquad \langle \sigma, \gamma, \Pi, R \rangle \models_B \beta \qquad \{\phi_1, \ldots, \phi_n, \phi\} \not\models \bot}{\langle \sigma, \gamma, \Pi, R \rangle \longrightarrow \langle \sigma, \gamma, \Pi \cup \{(\pi, \phi)\}, R \rangle}$$

Notice that the condition of consistency of the agent's intentions is maybe not yet the best option to avoid irrational actions, Winikoff et al. suggest, in [18], that it is necessary also to analyze the *plans* of the agent, as well as the *resources* available to achieve the intentions. However, this is out of the scope of this paper. The reader can also notice that the *conjunction of goals* cannot be expressed by only considering the intention adoption rule. It is necessary to increment the goal base of the agent. Consider that we want to program an agent with the following goal $a_1 \wedge, \ldots, \wedge a_n$. We can express this goal by having the following rules in the goal base $conj\_a_s \leftarrow a_1, \ldots, a_n$ and $conj\_a_s \leftarrow$, where $conj\_a_s$ is a new variable in the goal base. The goal $conj\_a_s$ will only be achieved if the conjunction $a_1 \wedge, \ldots, \wedge a_n$ is true.

We have just introduced a rule to adopt new intentions. Considering that intentions are committed goals, if the goal that the intention represents is no longer pursued by the agent, it would make sense to drop it. Therefore, we introduce into our agent framework the *Intention Dropping Rule*. Informally, the semantics of this rule is to remove from its intention base, any intention that is no longer supported by the goal base of an agent. The next definition formalizes this idea.

Definition 9 (Intention Dropping Rule). *Let $\langle \sigma, \gamma, \Pi, R \rangle$ be an agent configuration, where $\{(\pi, \phi)\} \subseteq \Pi$. Then:*

$$\frac{\langle \sigma, \gamma, \Pi, R \rangle \not\models_G G\phi}{\langle \sigma, \gamma, \Pi, R \rangle \longrightarrow \langle \sigma, \gamma, \Pi \setminus \{(\pi, \phi)\}, R \rangle}$$

As the intention dropping rule is defined, the agent could stop executing a plan if a goal is no longer entailed by the goal base. Stopping abruptly the execution of the plan could be undesired since there might be some *cleaning actions* to be taken after the goal is achieved. For example, if an agent's goal is to *bake a cake*, it would execute an appropriate plan, gathering the ingredients, the utensils, and setting up the oven. After the cake is baked the agent would still have to wash the utensils and throw the garbage away, these actions could be seen as clean up actions. To handle this issue, we could propose a more complex system of intentions, where there would be two plans associated with the committed goal, one used to achieve the goal and another used to do the *cleaning up*. When the goal is achieved the agent would execute the cleaning up plan. However, this issue is not our main interest here in this paper, and therefore we will limit our system to the intention dropping rule proposed in the definition above.

To be able to use the update semantics of DLP it is interesting to have a rule that can update the goal base of an agent with a generalized logic program. We will call this rule as *Goal Update Rule*. We will investigate in the Section 4, how to use the Goal Update Rule to adopt, drop or modify goals.

Definition 10 (Goal Update Rule). *Let P be a Generalized Logic Program and $\beta \in \mathcal{L_B}$ be a query formula. The Goal Update Rule is defined as the tuple, $\langle \beta, P \rangle$. We will call β as the precondition of the goal update rule.*

Informally, the *semantics* of the goal update rule $\langle \beta, P \rangle$, is that when the precondition, β, is satisfied the goal base of an agent is updated by the generalized logic program P.

Definition 11 (Semantics of Goal Update Rules). *Let $\langle \sigma, \gamma, \Pi, R \rangle$ be an agent configuration, the semantics of a Goal Update Rule, $\langle \beta, P \rangle \in R$ is given by the transition rule:*

$$\frac{\langle \sigma, \gamma, \Pi, R \rangle \models_B \beta}{\langle \sigma, \gamma, \Pi, R \rangle \longrightarrow \langle \sigma, (\gamma, P), \Pi, R \rangle}$$

5 Adopting and Dropping Goals

In this section we are going to investigate how to represent, in our system, situations where an agent has to adopt or drop goals. We begin, in Subsection 4.1, by investigating how to represent failure conditions for goals. We will also define, in this Subsection, how to represent maintenance and achievement goals, since they are important concepts to be analyzed by an agent when it is intending to drop a goal. Later, in Subsection 4.2, we discuss some possible motivations of why an agent should adopt a goal and also investigate how to represent these motivations in our agent framework. Finally in Subsection 4.3, we identify some further properties of our framework.

5.1 Goal Dropping

In this subsection, we are going to investigate some situations where the agent must *drop a goal* and discuss how this could be done with our agent framework.

Winikoff et al. in [18], suggests some properties that the agent should have with respect to its goals, one of these properties is being able to define *failure conditions*. The idea is that when the failure condition is true the goal should be dropped and, furthermore, the agent should remove it from its intention base in case it had committed to it.

We can easily define failure conditions for goals using Dynamic Logic Programs, since failure conditions can be viewed as conditional goals. Consider the following example.

Example 2. Consider an agent that has to write a paper until a deadline of a conference. We could represent this situation using the following DLP, composed by a single GLP with a single rule, $write\_paper \leftarrow not\ deadline\_over$. The agent will consider $write\_paper$ as a goal only if the deadline is not over.

Another situation where the agent should drop a goal (or an intention) is when the goal (or intention) has been achieved, i.e., when the belief base entails the goal (or intention). By Proposition 1, we have that the agent will never entail a goal formula that is believed to be achieved. Hence, the agent can use the Intention Dropping Rule to drop intentions that are no longer goals of the agent.

Up to now we haven't explored the full expressiveness of Dynamic Logic Programs, by the simple fact that we didn't need, in any of the examples, the update semantics of DLP. We are going to use the semantics of DLP to be able to construct agents that can have *maintenance* as well as *achievement* goals.

In what circumstances an agent should drop a goal will depend in which type of goal it is. If it is an achievement goal, once it is achieved the goal must be dropped and not pursued in the future anymore. And if it is a maintenance goal, it will only be dropped when it is currently entailed by the agent's beliefs. But if in the future the goal is no longer entailed by its belief base, the agent will have to pursue this goal once more.

To be able to differentiate between these types of goals, we are going to define a special predicate, only appearing in the goal base, with signature, *maintenance(.)*, stating that the goal as argument is a maintenance goal. The following definition makes this precise.

Definition 12 (Maintenance and Achievement Goals). *Let $\langle \sigma, \gamma, \Pi, R \rangle$ be an agent configuration. We will call the goal ϕ as a maintenance goal iff*

$$\langle \sigma, \gamma, \Pi, R \rangle \models_G Gmaintenance\,(\phi) \wedge \langle \sigma, \gamma, \Pi, R \rangle \models_G G\phi$$

We call the goal ϕ an achievement goal iff

$$\langle \sigma, \gamma, \Pi, R \rangle \not\models_G Gmaintenance\,(\phi) \wedge \langle \sigma, \gamma, \Pi, R \rangle \models_G G\phi$$

We are going to use the semantics of DLP to define a *goal update operator* that updates the goals of the agent by dropping the achievement goals that have been achieved. The idea is to apply the goal update operator whenever the belief base of the agent is changed (this could be done by a deliberation cycle).

Definition 13 (Goal Update Operator - Ω). *Let $\langle \sigma, \gamma, \Pi, R \rangle \longrightarrow \langle \sigma', \gamma', \Pi', R \rangle$ be a transition in the transition system, where $\langle \sigma, \gamma, \Pi, R \rangle$ and $\langle \sigma', \gamma', \Pi', R \rangle$ are agent configurations, and $\Gamma(\sigma) = \{not\ L \leftarrow not\ maintenance(L) \mid L \in \sigma\}$. We define the goal update operator, Ω, as follows:*

$$\Omega(\gamma, \sigma') = \gamma' = (\gamma, \Gamma(\sigma'))$$

We must be sure that with the *goal update operator* defined above, new goals are not created and only the goals that have to be dropped are removed from the Goal Models. The next theorem states that when the goal update operator is used, no achievement goals that are achieved will be entailed by the agent, regardless of its future beliefs. For example, consider that an agent has achieved a goal ϕ and has updated its goal base with the goal update operator. If the agent doesn't adopt ϕ as a goal once more, or changes its status to a maintenance goal, ϕ will not be a goal of the agent even if in the future, the agent's belief base doesn't entail ϕ.

Theorem 1. *Let $\langle \sigma, \gamma, \Pi, R \rangle$ be an agent configuration and σ' be another belief base, such that $\langle \sigma, \gamma, \Pi, R \rangle \models_G G\phi$ and $\sigma' \models \phi$. Then:*

$$(\forall \sigma'' \in \mathcal{I}).(\langle \sigma'', \gamma', \Pi, R \rangle \not\models Gmaintenance(\phi) \Rightarrow \langle \sigma'', \gamma', \Pi, R \rangle \not\models G\phi)$$

where $\gamma' = \Omega(\gamma, \sigma')$.

Proof. Proof: Since $\sigma' \models \phi$ the goal update operator will update γ with a rule r, $\{not\ \phi \leftarrow not\ maintenance(\phi)\}$. As $\langle \sigma'', \gamma', \Pi, R \rangle \not\models Gmaintenance(\phi)$, the rule r will be activated rejecting all the rules with head, ϕ. Hence $\langle \sigma'', \gamma', \Pi, R \rangle \not\models G\phi$.

By proposition 1 we have that the maintenance goals will not be entailed by the agent if it believes that it is currently achieved.

5.2 Goal Adoption

Agents often have to adopt new goals. The reasons for adopting new goals can be varied, the simplest one, when dealing with pro-active agents, would be because the agent doesn't have any goals and it is in an *idle* state.

We follow [16], and distinguish two motivations behind the adoption of a goal: *internal* and *external*. Goals that derive from the desires of the agent, represented by *abstract goals*, have an internal motivation to be adopted. External motivations, such as *norms*, *impositions* from other agents, and *obligations*, can also be a reason for the agent to adopt new goals. An example of a norm, in the daily life, is that a person should obey the law. Obligations could derive from a *negotiation* where an agent commits to give a service to another agent e.g. your internet provider should (is obliged to) provide the internet connection at your home. Agents usually have a *social point of view* e.g. a son usually respects his father more than a stranger, and it may be the case that an agent imposes another agent some specific goals e.g. a father telling the son to study.

Dignum and Conte discuss, in [5], that an agent usually has *abstract goals* that are usually not possible to be achieved by a simple plan, but the agent believes that these abstract goals can be approximated by a set of *concrete goals*. Notice that the beliefs of the agent must be taken in consideration to adopt new concrete goals. For example, if an agent has the *desire* to obey the law and it believes that if it drives too fast it will break the law, it might have the goal of driving slower. On the other hand, it would be *acceptable* for the agent to talk on the mobile phone while driving a car, if an agent believes that by doing so it is not breaking the law, even though, by doing so, it might be *violating* the law.

Using DLP as the goal base of an agent we can *partially* simulate this behavior. Consider an agent with the a goal base consisting of one GLP, $\{\neg drive\_fast \leftarrow obey\_law; obey\_law \leftarrow; maintenance(obey\_law) \leftarrow\}$. As the agent will have the abstract maintenance goal of obeying the law (however there might be no plan to achieve it), it will try not to drive fast.

To be able to commit to obligations, changes in norms, or changes in desires, we need to be able to change the goal base during execution. For example, if a new deal is agreed to provide a service to another agent, the agent must entail this new obligation. By using the Goal Update Rule, an agent can update its goal base in such a way that it can *incorporate* new goals in several situations:

Adopt New Concrete Goals - As discussed previously, the agent may have some desires that can be represented by abstract goal κ that is usually not really achievable, but the agent believes that it can be approximated by some concrete goals $(\kappa_1, \ldots, \kappa_n)$. Consider that the agent learns that there is another concrete goal κ_l that, if achieved, can better approximate the abstract goal, κ. The agent can update its goal base using the following Goal Update Rule, $\langle concrete\_goal(\kappa_l, \kappa), \{\kappa_l \leftarrow \kappa\}\rangle$, as κ is a goal of the agent, it will activate the new rule, hence the new concrete goal, κ_l, will also be a goal of the agent;

Norm Changes - Consider that the agent belongs to a *society* with some norms that have to be obeyed $(norm_1, \ldots, norm_n)$ and furthermore that there is a change in the norms. Specifically, the $norm_i$ is changed to $norm'_i$, hence the agent's goal base must change. We do this change straightforwardly, using the goal update rule, $\langle change(norm_i, norm'_i), \{not\ norm_i \leftarrow; norm'_i \leftarrow\}\rangle$. This update will force all the rules, r, with $Head(r) = norm_i$ to be rejected and $norm_i$ will no longer be a goal of the agent. Notice that there must be some coherence with the change in the norms. For example, the agent shouldn't believe that on $change(norm_i, norm_j)$ and at the same time on $change(norm_j, norm_i)$;

New Obligations - Agents are usually immersed with other agents in an environment and, to achieve certain goals, it might be necessary to negotiate with them. After a negotiation round, it is normal for agents to have an agreement that stipulates some conditions and obligations (e.g. in *Service Level Agreements* [7]). The agent can again easily use the goal update rules to incorporate new obligations, $\langle obligation(\phi)$, $\{\phi \leftarrow\}\rangle$, as well as *dismiss* an obligation when an agreement is over, $\langle \neg obligation(\phi)$, $\{not\ \phi \leftarrow\}\rangle$;

Impositions - Agents not only negotiate, but sometimes have to *cooperate* with or *obey* other superior agents. This sense of superiority is quite subjective and can be, for example, the obedience of an employee to his boss, or a provider towards his client. It will depend on the beliefs of the agent to decide if it should adopt a new goal or not, but this can be modeled using the goal update rule, $\langle received(achieve, \phi, agent_i) \wedge obey(agent_i)$, $\{\ \phi \leftarrow\}\rangle$. Meaning that if it received a message from $agent_i$ to adopt a new goal ϕ, and the receiving agent believes it should obey $agent_i$, it will update its goal base. Notice that more complex hierarchy could be achieved by means of *preferences* between the agents. However, it would be necessary to elaborate a mechanism to solve possible *conflicts* (e.g by using Multi-Dimensional Dynamic Logic Programming [10]).

5.3 Further Properties

We still can identify some more properties that could be elegantly achieved by using the goal update rule:

Defining Maintenance and Achievement Goals We can define a goal as a maintenance goal if a certain condition is satisfied. For example, an initially single male agent finds the woman agent of its life and marries it. After this is achieved, it might like to be married with this agent until the end of its life. This can be represented by the goal update rule $\langle married(girl)$, $\{$ $married(girl) \leftarrow; maintenance(married(girl)) \leftarrow\}\rangle$. The opposite can also be easily achieved, using the goal update rule. A goal that initially was a maintenance goal can be dropped or switched to an achievement goal. For example, consider that the previous agent had a fight with its agent wife and, after the divorce, it doesn't want to marry again. This can be represented by the goal update rule, $\langle divorce(girl)$, $\{\ not\ married(girl) \leftarrow$ $;\ not\ maintenance(married(girl)) \leftarrow\}\rangle$. We define a new achievement or modify a maintenance goal to an achievement by using the following goal update rule $\langle achieve(\phi)$, $\{\ \phi \leftarrow;\ not\ maintenance(\phi) \leftarrow\}\rangle$;

Defining and Modifying Failure Conditions and Conditional Goals - As discussed, failure conditions are used to define when a goal has to be dropped. It is possible that the agent is not aware of all the failure conditions for a goal, or there has been a change in the environment such that the previous failure is not enough or, furthermore, it is not a valid failure condition anymore. Using the goal update rule, we are able to define *new*, *modify* or even *eliminate* failure conditions. Consider the example where the agent has to write a paper until a deadline and the deadline is postponed, we can use the following goal update rule, $\langle postponed\_deadline, \{write\_paper \leftarrow postponed\_deadline\}\rangle$. Conditional goals can be defined using a similar goal update rule.

6 Example

Consider a scenario containing two agents, a *father* and a *son.* Furthermore consider that the father agent is the head of a *mob family.* The son agent wants to obey the law but only if by doing so he doesn't disobey its father. Obeying the law can be viewed as an abstract goal that will be approximated by more concrete goals. These concrete goals can also been seen as the norms that the society imposes on the son agent. However, according to his social viewpoint, his father is more important than the society itself.

The goal base of the son agent can be represented by the following DLP:

$$\begin{aligned} \neg kill &\leftarrow obey\_law, not\, disobey\_father. \\ disobey\_father &\leftarrow received\,(father, \phi, command)\,, not\, \phi. \\ \phi &\leftarrow received\,(father, \phi, command)\,. \\ obey\_law &\leftarrow . \\ maintenance\,(obey\_law) &\leftarrow . \end{aligned}$$

Considering an initially empty belief set, the son agent has a unique Goal Model, namely $\{maintenance(obey\_law), obey\_law, \neg kill\}$. Consider that, subsequently, his father orders him to kill one of the mobsters of the rival family. Hence, the son receives the achievement goal of killing, modifying its beliefs to $\{received(father, kill, command)\}$. Therefore, the Goal Model of the son agent changes to $\{maintenance\ (obey\_law),\ obey\_law, kill\}$[6].

The son agent, after committing to the goal of killing, will create a plan to achieve it and, after executing the plan (killing the mobster), the agent updates its goal base with the rule

$$not\, kill \leftarrow not\, maintenance\,(kill)\,.$$

And the Goal Model of the son is again $\{maintenance(obey\_law), obey\_law, \neg kill\}$. Consider now that the politicians, being annoyed by the gambling in city,

[6] Notice that $\neg kill$ is not in the Goal Model because we are using the expanded version of the GLPs

resolved to consider it illegal. Accordingly, the goal base of the son is updated with the GLP consisting of the following rule:

$$\neg gamble \leftarrow obey\_law, not\, disobey\_father.$$

The Goal Model of the son agent would change to $\{maintenance(obey\_law), obey\_law, \neg kill, \neg gamble\}$. However, his father, not being happy with this decision, orders his son to continue the gambling activities. Hence, $\{received(gamble, \phi, command)\}$ is added to his beliefs and the Goal Model of the son changes to

$$\{maintenance(obey\_law), obey\_law, \neg kill, gamble\}$$

This example illustrates how a programmer can use the Goal Update Rule to represent changes in the norms (considering gambling illegal) and use DLPs to represent concrete goals (not killing and not gambling). Furthermore, we could represent, in this small scenario, a social point of view of an agent (the son's social point of view) and how to give the *correct preference* on the goals according to this view.

7 Conclusions

In this paper, we introduced a simple agent framework with the purpose of introducing the agent's goal base as a Dynamic Logic Program. We investigated some properties of this framework. We were able to express, in a simple manner, conditional, maintenance and achievement goals, as well as identify some situations where the agent would need to adopt and drop goals, and how this could be done in this framework.

Since the objective of this paper was to investigate the use of DLP as the goal base of an agent, we didn't investigate what additional properties we could have by also using the belief base as a DLP. We also didn't give an adequate solution for conflicting intentions, since it would probably be also necessary to analyze the plans of the agent as well as its resources [18] to be able to conclude which goals to commit to.

Further investigation could also be done to solve possible conflicts in the social point of view of the agent. For example, if the agent considers the opinion of his mother and father equally, it would be necessary to have a mechanism to solve the conflicts since the agent doesn't prefer any one of them more than the other. [10] introduces the concept of *Multi Dimensional Dynamic Logic Programming* (MDLP) that could represent an agent's social point of view. Further investigation could be made in trying to incorporate the social point of view of an agent as a MDLP in our agent framework.

Even though this is still a preliminary report, we believe that DLP is a promising approach in which to represent the declarative goals of an agent, since it easily allows for the representation of the various aspects associated with agents' goals, and their updates, while enjoying a formal well defined semantics.

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), 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. M. Bratman. *Intentions, Plans and Practical Reason.* Harvard University Press, 1987.
4. 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*, chapter 2. Springer, 2005.
5. F. Dignum and R. Conte. Intentional agents and goal formation. In *Intelligent Agents IV*, volume 1365 of *LNAI*, pages 231–243, 1998.
6. K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming with declarative goals. In *Intelligent Agents VII*, volume 1986 of *LNAI*, pages 228–243. Springer, 2000.
7. N. R. Jennings, T. J. Norman, P. Faratin, P. O'Brien, and B. Odgers. Autonomous agents for business process management. *Applied Artificial Intelligence*, 14(2):145–189, 2000.
8. J. Leite. *Evolving Knowledge Bases.* IOS press, 2003.
9. J. Leite. On some differences between semantics of logic program updates. In *IBERAMIA'04*, volume 3315 of *LNAI*, pages 375–385. Springer, 2004.
10. J. Leite, J. J. Alferes, and L. M. Pereira. On the use of multi-dimensional dynamic logic programming to represent societal agents' viewpoints. In *EPIA'01*, volume 2258 of *LNAI*, pages 276–289. Springer, 2001.
11. 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.
12. J. Leite and L. M. Pereira. Generalizing updates: From models to programs. In *LPKR'97*, volume 1471 of *LNAI*, pages 224–246. Springer, 1998.
13. Á. F. Moreira, R. Vieira, and R. H. Bordini. Extending the operational semantics of a BDI agent-oriented programming language for introducing speech-act based communication. In *DALT'03*, volume 2990 of *LNAI*, pages 135–154. Springer, 2004.
14. V. Nigam and J. Leite. Incorporating knowledge updates in 3apl. In *PROMAS'06*, 2006.
15. J. Thangarajah, L. Padgham, and M. Winikoff. Detecting & avoiding interference between goals in intelligent agents. In *IJCAI'03*, pages 721–726. Morgan Kaufmann, 2003.
16. B. van Riemsdijk, M. Dastani, F. Dignum, and J.-J. Ch. Meyer. Dynamics of declarative goals in agent programming. In *DALT'04*, volume 3476 of *LNAI*, pages 1–18, 2004.
17. M. B. van Riemsdijk, M. Dastani, and J.-J. Ch. Meyer. Semantics of declarative goals in agent programming. In *AAMAS'05*. ACM Press, 2005.
18. M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah. Declarative and procedural goals in intelligent agent systems. In *KR'02*. Morgan Kaufmann, 2002.
19. M. Wooldridge. *Multi-agent systems : an introduction.* Wiley, 2001.