Defeasible Argumentation Support for an Extended BDI Architecture[*]

Nicolás D. Rotstein, Alejandro J. García, and Guillermo R. Simari

Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
Artificial Intelligence Research and Development Laboratory
Department of Computer Science and Engineering
Universidad Nacional del Sur, Bahía Blanca, Argentina
{ndr,ajg,grs}@cs.uns.edu.ar

Abstract. In this work, an agent architecture that combines defeasible argumentation and the BDI model is described. Argumentation will be used as a mechanism for reasoning about beliefs, for filtering desires considering the agent's current environment, and for selecting proper intentions. The approach allows to define different types of agents and this will affect the way in which desires are filtered and hence, which intention is selected. For performing defeasible reasoning, the approach uses a concrete framework based on a working defeasible argumentation system: Defeasible Logic Programming (DeLP). A set of filtering rules, represented as a defeasible logic program, will be used to represent reasons for and against adopting desires. Thus, based on its perceived or derived beliefs, the agent will argue about which of its desires are achievable in the current situation. To clarify the ideas two applications will be introduced to show two significantly different types of agent that can be implemented using this approach.

1 Introduction and Motivation

In this work, an agent architecture that combines defeasible argumentation and the BDI model is described. Argumentation will be used for reasoning about beliefs, for filtering desires considering the agent's current environment, and for selecting proper intentions. The approach allows to define different types of agents and this will affect the way in which desires are filtered and hence, which intention is selected. For performing defeasible reasoning, the approach uses a concrete framework based on a working defeasible argumentation system: Defeasible Logic Programming (DeLP).

This work is an extension of the article "Reasoning from Desires to Intentions: A Dialectical Framework" published in AAAI 2007 by the same authors [1]. Here, besides presenting the approach, we focus on two types of applications: a security system and robotic soccer. They were chosen because they represent two significantly different kinds of agents that can be implemented using our approach. As explained below, the security-system agent will have the goal of handling unexpected problematic situations, whereas the soccer agent will control the behavior of a robot in order to play somehow successfully.

[*] Partially supported by CONICET, ANPCyT, and UNSur.

In the first application domain, we consider a security system agent supervising a building with several rooms (*e.g.*, a museum). The agent's goal will be to act in case of unexpected problems (*e.g.*, fire or intruders) and to decide which is the best action to take in each case. The agent will have sources of information from which to gather beliefs: video cameras and smoke, motion and temperature sensors. These sensors can be thought as coupled and thus work as two mutual backup subsystems: the smoke/temperature sensors pair would detect fire and the camera/motion sensor pair, intruders. As it will be explained in detail in Section 7, agent's intentions could be: *"send a guard to a room"*, "*call the police*" or "*call the firemen*".

The security system agent perceives information from the environment through the mentioned sensors and this information represents its *perceived belief.* (*e.g.*, *"there is motion in room 2"* or *" there is no smoke in room 3"*). Besides perceived beliefs the agent may have more knowledge represented as a defeasible logic program (Section 2) that will be used for warranting *derived beliefs*.

Our approach provides a defeasible reasoning mechanism for filtering agent's desires in order to obtain a set of current desires, *i.e.*, those that are achievable in the current situation. A set of *filtering rules*, represented as a defeasible logic program, will be used to represent reasons for and against adopting desires. For example, the defeasible rule $call(firemen) \mathrel{-\!\!\prec} smoke(R)$ means "if there is smoke in a room R then there are reasons for calling the firemen". Thus, the security system agent will be provided with a set of filtering rules that will represent reasons for and against adopting one of its desires, *i.e.*, *call the firemen*, *call the police* or *send a guard*. Thus, based on its perceived or derived beliefs, the agent will debate which of its desires are achievable in the current situation. For example, if the agent perceive that there is smoke in one room then "*call firemen*" could be one of its *current desires*. Since the approach allows to define different agent types, in the case of the security system application we will develop a cautious agent, that is, an agent that only selects warranted desires. Once the set of current desires is obtained, then the agent will be able to select one intention. The security system agent will be explained in detail in Section 7.

The other application domain that we will consider in this work is robotic soccer. Our robotic soccer agent senses its environment through a video camera that takes the whole playing field, and from that perception it can build its set of perceived beliefs (*e.g.*, *it is marked, a mate has the ball*). Our proposed agent will have rules in order to derived other beliefs, for instance, the defeasible rule "*if a mate has the ball then the agent may receive the ball*" will allow the agent to build an argument for the belief that it may receive the ball, based on the perception that a mate has the ball. However, as we will show in detail in the next section, other rule like "*if the agent is marked and a mate has the ball then it will not receive the ball*" can be used for building a counter-argument for the previous one. The set of desires of the soccer agent could be *shoot, carry, pass* and *move*, *i.e.*, shoot to goal, carry the ball, pass the ball to a teammate and move to a different position in the field.

A significant difference between the two application domains is how they select intentions. The security-system agent is allowed to select and fulfill possibly many intentions at the same time, because it would have to deal with multiple hazardous situations simultaneously. In opposition to this, a robotic-soccer agent can pursue just one intention at a time, since, for instance, it cannot shoot on goal and pass the ball to a teammate at once.

2 The Proposed Architecture

An outline of this architecture appears in Fig. 1 [1]. Briefly, the main input is the *perception* from the environment, which is part of the set of *belief rules* (Π_{B},Δ_{B}) that, through an *argumentation process*, leads to the set B of warranted beliefs. For example, suppose that a soccer agent perceives that *it is marked* and *a teammate has the ball*, then it can warrant the belief *"I will not receive the ball"*.

As shown in the figure, the set of filtering rules, along with a set D of desires and the specification of a *filtering function* are the input to a *dialectical filtering process*, whose output is the set D^c of the agent's *current desires*. Following our example, consider that our soccer agent has the filtering rule "*if I will not receive the ball then there is a reason to move to a different place*". Since there is a warrant for "*I will not receive the ball*", then *move* will be a current desire. The final stage of the *agent behavior loop* shown in the figure involves the usage of a set of *intention rules*, embedded in an *intention policy* that will determine the preferred rule. The current desire in the head of this rule will be the *selected intention*.

As shown in Fig. 1, there are three main processes. They use defeasible argumentation based on Defeasible Logic Programming (DeLP). Next, we give a brief summary of DeLP (for more details see [2]). In DeLP, knowledge is represented using facts, strict rules, and defeasible rules:

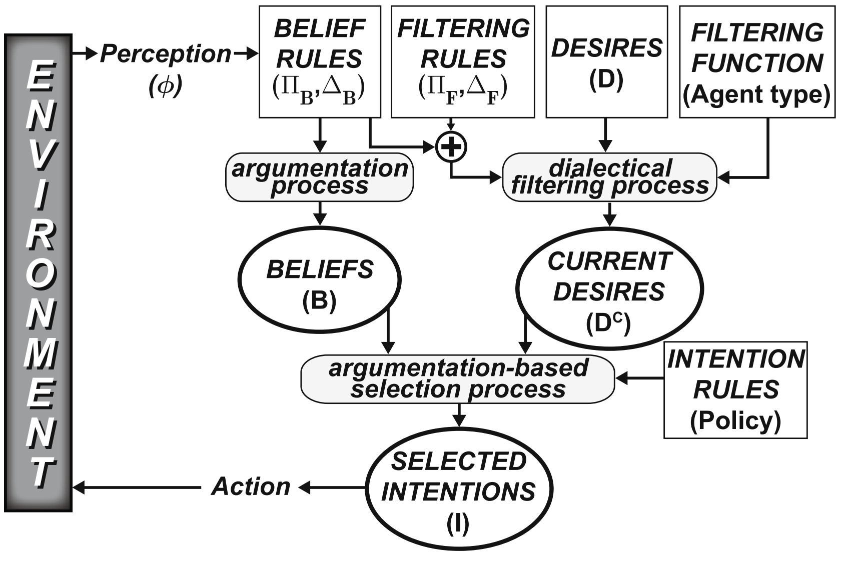

Fig. 1. DeLP-based BDI architecture

- *Facts* are ground literals representing atomic information or the negation of atomic information using strong negation "$\sim$" (*e.g.*, $hasBall(opponent)$).
- *Strict Rules* are denoted $L_0 \leftarrow L_1, \ldots, L_n$, where L_0 is a ground literal and $\{L_i\}_{i>0}$ is a set of ground literals (*e.g.*, $\sim hasBall(myTeam) \leftarrow hasBall(opponent)$).
- *Defeasible Rules* are denoted $L_0 \prec L_1, \ldots, L_n$, where L_0 is a ground literal and $\{L_i\}_{i>0}$ is a set of ground literals. (*e.g.*, $\sim pass(mate1) \prec marked(mate1)$).

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

A Defeasible Logic Program (*de.l.p.*) $\mathcal{P}$ is a set of facts, strict rules and defeasible rules. When required, $\mathcal{P}$ is denoted (Π, Δ) distinguishing the subset Π of facts and strict rules, and the subset Δ of defeasible rules. Strict and defeasible rules are ground, however, following the usual convention [3], some examples will use "schematic rules" with variables.

Strong negation could appear in the head of program rules, and can be used to represent contradictory knowledge. From a program (Π, Δ) contradictory literals could be derived, however, the set Π (used to represent non-defeasible information) must be non-contradictory, *i.e.*, no pair of contradictory literals can be derived from Π. Given a literal L, $\overline{L}$ represents the complement with respect to strong negation. If contradictory literals are derived from (Π, Δ), a dialectical process is used for deciding which literal prevails. In short, an *argument* for a literal L, denoted $\langle \mathcal{A}, L \rangle$, is a minimal set of defeasible rules $\mathcal{A} \subseteq \Delta$, such that $\mathcal{A} \cup \Pi$ is non-contradictory, and there is a derivation for L from $\mathcal{A} \cup \Pi$. A literal L is *warranted* from (Π, Δ) if there exists a non-defeated argument $\mathcal{A}$ supporting L. To establish if $\langle \mathcal{A}, L \rangle$ is a non-defeated argument, *argument rebuttals* or *counter-arguments* that could be *defeaters* for $\langle \mathcal{A}, L \rangle$ are considered, *i.e.*, counter-arguments that by some criterion are preferred to $\langle \mathcal{A}, L \rangle$. A defeater $\mathcal{A}_1$ for an argument $\mathcal{A}_2$ can be proper ($\mathcal{A}_1$ *stronger than* $\mathcal{A}_2$) or *blocking* (same strength). In the examples that follow we assume generalized specificity as the comparison criterion, however, as explained in [2] the criterion could be easily changed.

Since defeaters are arguments, there may exist defeaters for them, and defeaters for these defeaters, and so on. Thus, a sequence of arguments called *argumentation line* is constructed, where each argument defeats its predecessor in the line (for a detailed explanation of this dialectical process see [2]). In DeLP, a query Q could have four possible answers: YES, if Q is warranted; NO, if the complement of Q is warranted; UNDECIDED, if neither Q nor its complement is warranted; and UNKNOWN, if Q is not in the signature of the program.

3 Warranting Beliefs

Following [4], agent's beliefs correspond to the semantics[1] of a *defeasible logic program* $\mathcal{P}_B = (\Pi_B, \Delta_B)$. In Π_B two disjoint subsets will be distinguished: Φ of *perceived beliefs* that will be updated dynamically (see Fig. 1), and Σ of strict rules and facts that

[1] Since the semantics of DeLP is skeptical, there is only one.

will represent static knowledge, $\Pi_{\mathsf{B}} = \Phi \cup \Sigma$. Besides the perceived beliefs, the agent may use strict and defeasible rules from $\mathcal{P}_{\mathsf{B}}$ to obtain a *warrant* for its *derived beliefs* (see Definition 1).

We require Π_{B} to be non-contradictory, and also assume that perception is correct in the sense that it will never give a pair of contradictory literals. The next definition introduces the different types of belief that an agent will obtain from a defeasible logic program (Π_{B},Δ_{B}).

Definition 1 (Belief types). *A **Perceived belief** is a fact in Φ that the agent has perceived directly from its environment. A **Strict belief** is a literal that is not a perceived belief, and it is derived from $\Pi_{\mathsf{B}} = \Phi \cup \Sigma$ (i.e., no defeasible rules are used for its derivation). A **Defeasible belief** is a warranted literal L supported by an non-empty argument $\mathcal{A}$ (i.e., it uses at least one defeasible rule). Finally, a **Derived belief** is a strict or a defeasible belief. We will denote with B_s the set of strict beliefs, and with B_d the set of defeasible beliefs. Therefore, in any given situation, the beliefs of an agent will be $\mathsf{B} = \Phi \cup \mathsf{B}_s \cup \mathsf{B}_d$.*

Example 1. *Consider a robotic-soccer agent with the following program (Π_{B},Δ_{B}), where Π_{B} was divided distinguishing the set Φ=$\{hasBall(t1),\ marked(t1)\}$ of perceived facts representing "player $t1$ has the ball", and "teammate $t1$ is marked", the set Σ of non-perceived information, and the set Δ_{B} of defeasible knowledge:*

$$\Sigma = \left\{ \begin{array}{l} mate(t1),\ opponent(o1), \\ (\sim mate(X) \leftarrow opponent(X)), \\ (\sim receive(self) \leftarrow hasBall(self)) \end{array} \right\}$$

$$\Delta_{\mathsf{B}} = \left\{ \begin{array}{l} (receive(self) \prec hasBall(X), mate(X)), \\ (\sim receive(self) \prec marked(self)), \\ (\sim receive(self) \prec hasBall(X), \sim mate(X)) \end{array} \right\}$$

From (Π_{B},Δ_{B}) the agent can infer the strict belief: $\sim mate(o1)$. The argument built from (Π_{B},Δ_{B}) for $receive(self)$: $\{receive(self) \prec hasBall(t1), mate(t1)\}$, has no defeaters, and therefore, there is a warrant for one defeasible belief: $receive(self)$ (the agent may receive a pass).

The sets Φ, B_s and B_d are disjoint sets. It can be shown that the set B of beliefs of an agent is a non-contradictory set of warranted literals. Although perceived beliefs are facts in Π_{B}, there could be other facts in Π_{B} which are not perceived, for instance, facts that represent agent's features, roles, *etc.* These facts that do not represent perceived information are persistent in the sense that they cannot change with perception, like $myRole(defender)$, or $mate(t1)$.

We assume a perception function that provides the agent with information about its environment. This function will be invoked by the agent to update its perceived beliefs set Φ. When this happens the new information overrides the old one following some criterion. Updating a set of literals is a well-known problem and many proposals exist in the literature [5,6]. Since we require Π_{B} to be non-contradictory, when Φ is updated, a revision function will ensure that Π_{B} remains a non-contradictory set. The specification of a proper revision operator is out of the scope of this paper.

Example 2. *In the context of Ex. 1, with the perception that the agent is now marked, the set Φ becomes $\{hasBall(t1), marked(t1), marked(self)\}$. Now the argument for $receive(self)$ has a "blocking defeater", which means that the DeLP answer for both $receive(self)$ and $\sim receive(self)$ will be* UNDECIDED.

Consider a different situation, where the perception is $\Phi = \{hasBall(o1)\}$. Here, the answer for $receive(self)$ is NO, *since there is a warrant for $\sim receive(self)$ supported by the non-defeated argument $\{\sim receive(self) \prec hasBall(o1), \sim mate(o1)\}$.*

4 Filtering Desires

Agents desires will be represented by a given set D of literals that will contain a literal representing each desire the agent might want to achieve. Clearly, D may be contradictory, that is, both a literal L and its complement $\overline{L}$ might belong to D. We will assume that beliefs and desires are represented with separate names, *i.e.*, $\mathsf{D} \cap \mathsf{B} = \emptyset$. Hence, a desire cannot be perceived or derived as a belief.

Set D represents all the desires that the agent may want to achieve. However, depending on the situation in which it is involved, there could be some desires impossible to be carried out. For example, if the agent does not have the ball and the ball is in a place p, then, the desire $shoot$ could not be effected, whereas $goto(p)$ is a plausible option. Therefore, agents should reason about their desires to select the ones that could be actually realized. Following the spirit of the BDI model, once appropriate desires are detected, the agent may select (and commit to) a specific intention (goal), and then select appropriate actions to fulfill that intention (see Figure 1).

In [4] a reasoning formalism was introduced for selecting from D those desires that are suitable to be brought about. To perform this selection, the agent uses its beliefs (representing the current situation) and a defeasible logic program (Π_F, Δ_F) composed by *filtering rules*. The filtering rules represent reasons for and against adopting desires. In other words, filtering rules eliminate those desires that cannot be effected in the situation at hand. Once the set of achievable desires is obtained, the agent can adopt one of them as an intention.

Definition 2 (Filtering rule). *Let D be the set of desires of an agent, a* filtering rule *is a strict or defeasible rule that has a literal $L \in \mathsf{D}$ in its head and a non-empty body.*

Observe that a filtering rule can be either strict or defeasible and, as will be explained below, that will influence the filtering process. Note also that a filtering rule cannot be a single literal (*i.e.*, a fact). Below we will explain how to use filtering rules in order to select desires, but first we will introduce an example to provide some motivation.

Example 3. *A robotic-soccer agent A^r could have the following sets of desires and filtering rules:*

$$\mathsf{D} = \left\{ \begin{array}{c} shoot \\ carry \\ pass \\ move \end{array} \right\} \quad \Pi_F = \left\{ \begin{array}{l} \sim carry \leftarrow \sim ball \\ \sim shoot \leftarrow \sim ball \\ \sim pass \leftarrow \sim ball \end{array} \right\}$$

$$\Delta_F = \left\{ \begin{array}{l} shoot \mathrel{-\!\!\prec} theirGoalieAway \\ carry \mathrel{-\!\!\prec} noOneAhead \\ pass \mathrel{-\!\!\prec} freeTeammate \\ \sim shoot \mathrel{-\!\!\prec} farFromGoal \\ \sim carry \mathrel{-\!\!\prec} shoot \\ move \mathrel{-\!\!\prec} \sim ball \end{array} \right\}$$

Consider a particular situation in which an agent does not have the ball (*i.e.*, $\sim ball \in \Phi$). If the agent has $\Delta_{\mathsf{B}} = \emptyset$, $\Pi_{\mathsf{B}} = \Phi$ and the filtering rules (Π_F,Δ_F) from Ex. 3, then, there are warrants for $\sim carry$, $\sim pass$ and $\sim shoot$ from this information. Hence, in this particular situation, the agent should not consider selecting the desires $carry$, $pass$, and $shoot$, because there are justified reasons against them. Observe that these reasons are not defeasible.

Consider now a different situation with a new set of perceived beliefs: $\mathsf{B} = \Phi = \{ball, theirGoalieAway, farFromGoal\}$, that is, a situation in which the agent has the ball and the opponent goalie is away from its position, but the agent is far from the goal. Then, from the agent's beliefs and the filtering rules (Π_F,Δ_F) of Ex. 3, there are arguments for both $shoot$ and $\sim shoot$. Since these two arguments defeat each other, a blocking situation occurs and the answer for both literals is UNDECIDED. In our approach (as will be explained later) an undecided desire could be eligible.

In this formalism, beliefs and filtering rules should be used in combination. Hence, we need to explain how two defeasible logic programs can be properly combined. Agents will have a *de.l.p.* $(\Pi_{\mathsf{B}},\Delta_{\mathsf{B}})$ containing rules and facts for deriving beliefs, and a *de.l.p.* (Π_F,Δ_F) with filtering rules for selecting desires. We need to combine these two *de.l.p.*, but the union of them might not be a *de.l.p.*, because the union of the sets of strict rules could be contradictory. To overcome this issue, we use a merge revision operator "$\circ$" [6]. Hence, in our case, the join of $(\Pi_{\mathsf{B}},\Delta_{\mathsf{B}})$ and (Π_F,Δ_F) will be a program (Π,Δ), where $\Pi = \Pi_{\mathsf{B}} \circ \Pi_F$ and $\Delta = \Delta_{\mathsf{B}} \cup \Delta_F \cup \Delta_X$. A set X is introduced, containing those strict rules r_i that derive complementary literals. This set is eliminated when merging Π_{B} and Π_F, then every r_i is transformed into a defeasible rule, and the set Δ_X is generated, carrying the resulting defeasible rules (see [4] for more details).

Definition 3 (Agent's Knowledge Base)
Let $(\Pi_{\mathsf{B}},\Delta_{\mathsf{B}})$ *be the set containing rules and facts for deriving beliefs;* (Π_F,Δ_F)*, the set of filtering rules; and* $\Delta_X = \{(\alpha \mathrel{-\!\!\prec} \gamma) \mid (\alpha \leftarrow \gamma) \in (\Pi_{\mathsf{B}} \cup \Pi_F)$ *and* $(\Pi_{\mathsf{B}} \cup \Pi_F) \vdash \{\alpha, \overline{\alpha}\}\}$*. Then* $K_{Ag} = (\Pi_{\mathsf{B}} \circ \Pi_F,\ \Delta_{\mathsf{B}} \cup \Delta_F \cup \Delta_X)$ *will be the agent's knowledge base.*

The next definition introduces a mechanism for filtering D obtaining only those desires that are achievable in the *current* situation. We allow the representation of different *agent types*, each of which will specify a different filtering process.

Definition 4 (Current desires). *Let* T *be a boolean function representing a selection criterion. The set* D^c *of* Current Desires *is defined as:*

$$\mathsf{D}^c = filter(T, \mathsf{D}) = \{\delta \in \mathsf{D} \mid T(\delta, K_{Ag}) = true\}.$$

Observe that the filtering function can be defined in a modular way. Methodologically, it would be important to make this function related to the K_{Ag}, in order to obtain a rational filtering. Implementing a sensible filtering function is not a trivial task, as it is domain-dependent, and a general criterion cannot be stated. Different agent types or personalities can be obtained depending on the chosen selection criterion T. The following are interesting alternatives:

- CAUTIOUS AGENT: $T(\delta, K_{Ag})$ is true when there is a warrant for δ from K_{Ag}.
- BOLD AGENT: $T(\delta, K_{Ag})$ is true when there is no warrant for $\overline{\delta}$ from K_{Ag}.

Notice that when neither δ nor $\overline{\delta}$ has a warrant built from K_{Ag}, then both literals will be included into the set D^c of a bold agent. Therefore, the agent will consider these two options (among others), albeit in contradiction.

The way a bold agent selects its current desires (see Ex. 4) becomes clearer considering the relation of warrant states with DeLP answers. In DeLP, given a literal Q, there are four possible answers for the query Q: YES, NO, UNDECIDED, and UNKNOWN. Thus, agent types using DeLP can be defined as follows:

- CAUTIOUS AGENT: $T(\delta, K_{Ag})$ is true when the answer for δ from K_{Ag} is YES.
- BOLD AGENT: $T(\delta, K_{Ag})$ is true when the answer for δ from K_{Ag} is YES, UNDECIDED or UNKNOWN.

Example 4. *Extending Ex. 3, if we consider a bold agent as defined above and the set of beliefs:*
$\mathsf{B} = \Phi = \{\, farFromGoal,\ noOneAhead,\ ball \,\}$
the agent will generate the following set of current desires:
$\mathsf{D}^c = \{carry, pass\}$
In this case, we have $K_{Ag} = (\Phi \circ \Pi_F, \emptyset \cup \Delta_F \cup \emptyset)$. Regarding D^c, DeLP's answer for $shoot$ is NO*, for $carry$ is* YES*, and for $pass$ is* UNDECIDED*. Finally, note that a cautious agent would choose $carry$ as the only current desire.*

As stated above, it is required that B and D be two separate sets to avoid the confusion when joining the (Π_{B},Δ_{B}) and (Π_F,Δ_F) programs. This is not a strong restriction, because a literal being both a belief and a desire brings about well-known representational issues, *e.g.*, symbol overload.

5 Selecting Intentions

In our approach, an intention will be a current desire $d \in \mathsf{D}^c$ that the agent can commit. To specify under what conditions the intention could be achieved, the agent will be provided with a set of *intention rules*. Next, these concepts and the formal notion of *applicable intention rule* are introduced.

Definition 5 (Intention Rule)
An intention rule *is a device used to specify under what conditions an intention could be effected. It will be denoted as $(d \Leftarrow \{p_1, \ldots, p_n\}, \{not\ c_1, \ldots, not\ c_m\})$, where d is a literal representing a desire that could be selected as an intention, $p_1, \ldots, p_n$ ($n \geq 0$) are literals representing preconditions, and $c_1, \ldots, c_m$ ($m \geq 0$) are literals representing constraints.*

Example 5. *The robotic-soccer agent A^r might have the following set of intention rules:*

$$\begin{array}{l} IR_1: (carry \Leftarrow \{ball\}, \{\}) \\ IR_2: (pass \Leftarrow \{ball\}, \{not\ shoot\}) \\ IR_3: (shoot \Leftarrow \{ball\}, \{not\ marked\}) \\ IR_4: (carry \Leftarrow \{winning\}, \{\}) \\ IR_5: (move \Leftarrow \{\}, \{\}) \end{array}$$

Now we describe how an intention becomes applicable.

Definition 6 (Applicable Intention Rule)
Let K_{Ag}= ($\Pi_{\mathsf{B}} \circ \Pi_F$, $\Delta_{\mathsf{B}} \cup \Delta_F \cup \Delta_X$) be the knowledge base of an agent, and D^c, its set of current desires. Let B be the set of beliefs obtained from (Π_{B},Δ_{B}). An intention rule $(d \Leftarrow \{p_1, \ldots, p_n\}, \{not\ c_1, \ldots, not\ c_m\})$ is applicable *iff*

1. *$d \in \mathsf{D}^c$,*
2. *for each precondition p_i $(0 \leq i \leq n)$ it holds $p_i \in (\mathsf{B} \cup \mathsf{D}^c)$*
3. *for each constraint c_i $(0 \leq i \leq m)$ it holds $c_j \notin (\mathsf{B} \cup \mathsf{D}^c)$.*

Thus, in every applicable intention rule it holds:

1. the head d is a current desire of the agent selected by the filtering function,
2. every precondition p_i that is a belief is warranted from K_{Ag},
3. every precondition p_i that is a desire belongs to set D^c,
4. every belief constraint c_i has no warrant from K_{Ag}, and
5. every c_i that is a desire does not belong to D^c.

Example 6. *Consider a bold agent, and K, B and D^c as given in Example 4. Now it is possible to determine which of the intention rules of Example 5 are applicable. Rule IR_1 is applicable because $carry \in \mathsf{D}^c$. Rule IR_2 is applicable because $pass \in \mathsf{D}^c$, $ball \in \mathsf{B}$, and $shoot \notin \mathsf{D}^c$. Rule IR_3 is not applicable because $shoot \notin \mathsf{D}^c$. Rule IR_4 is not applicable because the precondition is not a literal from K. Finally, IR_5 is not applicable because $move \notin \mathsf{D}^c$. Thus, $\{IR_1, IR_2\}$ is the set of applicable rules.*

Intention rules' goal is to select the final set of intentions. In general, this selection among current desires cannot be done by using filtering rules. For instance, if we have to select just one intention, and there are two warranted current desires, how can we choose one? There is a need for an external mechanism to make that decision.

Intention rules and filtering rules (Definition 2) have different semantics and usage:

– Filtering rules are used to build arguments for and against desires (thus, they are the basis of the dialectical process for warranting a desire), whereas intention rules are used on top of the dialectical process.
– Intention rules do not interact, whereas filtering rules do interact because they can be in conflict or can be used for deriving a literal in the body of another filtering rule.

– Applicable intention rules depend on the result of the filtering process over desires and warranted beliefs, whereas a filtering rule is "applicable" when its body literals are supported by perceived beliefs, or by other defeasible or strict rules.

The set of all applicable intention rules contains rules whose heads represent *applicable intentions* achievable in the current situation. Depending on the application domain, there are many possible policies to select from the set of applicable intentions. For example, the agent could try to pursue some of them simultaneously, or it might be forced to commit to one. Furthermore, each of these two options has, in turn, several solutions. The idea behind having intention rules and policies is to give a more flexible mechanism than plain priorities. Next, we define how to obtain a set of selected intentions.

Definition 7 (Set of Selected Intentions)
Let IR be the set of intention rules, and $App \subseteq IR$ be the set of all the applicable intention rules. Let $p : IR \rightarrow \mathsf{D}$ be a given selection policy. Then, the set of selected intentions I *will be $p(App)$.*

The policy $p(App)$ could be defined in many ways. For instance, $p(App)$ could be "return all the heads of rules in *App*". However, depending on the application domain, more restrictive definitions for $p(App)$ could be necessary. For example, in our robotic soccer domain, agents must select a single applicable intention at a time (*i.e.*, an agent cannot shoot and pass the ball at the same time). One possibility for defining a policy that returns a single intention is to provide a sequence with all the intention rules $[IR_1,...,IR_n]$ that represents a preference order among them. Then, the policy $p(App)$ selects the first rule IR_k ($1 \leq k \leq n$) in the sequence that belongs to *App*, returning the head of IR_k.

Example 7. *Continuing with Ex. 6. The set of applicable intention rules is $App = \{IR_1, IR_2, IR_5\}$, and suppose that the policy p is the one introduced above. Then, if the preference order is $[IR_1, IR_2, IR_3, IR_4, IR_5]$, the selected intention will be the head of IR_1, i.e., $p(App) = \{carry\}$.*

Now we can formally define the structure of an agent.

Definition 8 (DeLP-Based BDI Agent)
An agent A is a tuple $\langle \mathsf{D}, (\Pi_{\mathsf{B}},\Delta_{\mathsf{B}}), (\Pi_F,\Delta_F), T, IR, p\rangle$, where: D is the set of desires of the agent, $(\Pi_{\mathsf{B}},\Delta_{\mathsf{B}})$ is the agent knowledge (that will include perceived beliefs), (Π_F,Δ_F) are filtering rules, T is an agent type, IR is a set of intention rules, and $p(\cdot)$ is a policy for selecting intentions.

6 Application Example: Robotic Soccer

In this section a robotic-soccer agent A^r will be introduced and then we will show, using different examples, how A^r selects appropriate intentions when faced with different scenarios. In each example, the difference of defining a bold or a cautious agent will be made clear.

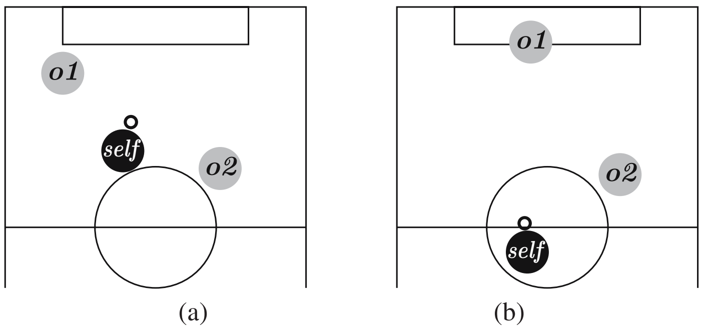

Fig. 2. Two scenarios for a robotic soccer agent

The robotic-soccer agent will be $A^r{=}\langle \mathsf{D}, (\Pi_{\mathsf{B}},\Delta_{\mathsf{B}}), (\Pi_F,\Delta_F), T, IR, p\rangle$ where the set D and (Π_F,Δ_F) are the ones from Ex. 3, the set IR is the one defined in Ex. 5, the policy p was defined in Ex. 7, and the set $\Delta_{\mathsf{B}} = \emptyset$.

Example 8. *Consider the agent A^r and the situation depicted in Fig. 2(a) where "o1" and "o2" represent the positions of two opponents and "self" is the position of the agent A^r who has the ball (small circle).*
Here, the perception of A^r is $\Phi_1 = \{ball, noOneAhead, theirGoalieAway\}$. In this situation, A^r can build the following arguments:
$\mathcal{A}_1 : \{shoot \mathrel{-\!\!\prec} theirGoalieAway\}$,
$\mathcal{A}_2 : \{carry \mathrel{-\!\!\prec} noOneAhead\}$,
$\mathcal{A}_3 : \{(\sim carry \mathrel{-\!\!\prec} shoot), (shoot \mathrel{-\!\!\prec} theirGoalieAway)\}$.
*Hence, shoot is warranted, whereas carry, ∼carry, pass and ∼pass are not. As stated above the filter function will determine the type of agent (*e.g.*, bold or cautious), which could affect the set of selected intentions. For example:*

- *for a cautious agent, $\mathsf{D}^c_{C1} = \{shoot\}$, intention rule IR_3 is applicable, and $\mathsf{I}_{C1} = \{shoot\}$;*
- *for a bold agent, $\mathsf{D}^c_{B1} = \{shoot, carry, pass\}$, intention rules IR_1 and IR_3 are applicable, and $\mathsf{I}_{B1} = \{carry\}$.*

Note that the cautious agent obtains only one current desire that is its selected intention. On the other hand, since the bold agent includes "undecided" literals in its current desires, D^c_{B1} has more elements than D^c_{C1}, there are two applicable intention rules, and the policy "p" has to be used.

Example 9. *Consider the agent A^r but in a different scenario (depicted in Fig. 2(b)). The perception of the agent is here $\Phi_2 = \{ball, noOneAhead, farFromGoal\}$. In this situation, A^r can build the following arguments:*
$\mathcal{A}_4 : \{\sim shoot \mathrel{-\!\!\prec} farFromGoal\}$,
$\mathcal{A}_5 : \{carry \mathrel{-\!\!\prec} noOneAhead\}$.
Hence, ∼shoot and $carry$ are warranted, whereas $pass$ and ∼$pass$ are not, and:

- *for a cautious agent, $\mathsf{D}^c_{C2} = \{carry\}$, intention rule IR_1 is applicable, and $\mathsf{I}_{C2} = \{carry\}$;*

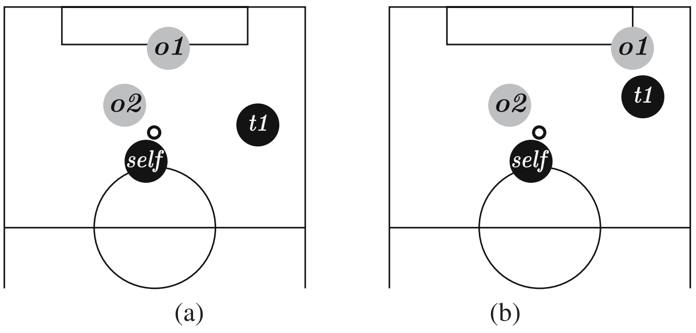

Fig. 3. Two scenarios for a robotic soccer agent

- *for a bold agent,* $\mathsf{D}^c_{B2} = \{carry, pass\}$*, intention rules* IR_1 *and* IR_2 *are applicable, and* $\mathsf{I}_{B2} = \{carry\}$.

Example 10. *Consider now that* A^r *is in the situation depicted in Fig. 3(a), where "t1" represents the position of a teammate of* A^r*. The perception of* A^r *is* $\Phi_3 = \{ball, freeTeammate, farFromGoal\}$*. In this situation,* A^r *can build the following arguments:*
$\mathcal{A}_6 : \{{\sim}shoot \prec farFromGoal\}$,
$\mathcal{A}_7 : \{pass \prec freeTeammate\}$,
Hence, we have that $pass$ *and* ${\sim}shoot$ *are warranted, whereas* $carry$ *and* ${\sim}carry$ *are not, and:*

- *for a cautious agent,* $\mathsf{D}^c_{C3} = \{pass\}$*, intention rule* IR_2 *is applicable, and* $\mathsf{I}_{C3} = \{pass\}$*;*
- *for a bold agent,* $\mathsf{D}^c_{B3} = \{carry, pass\}$*, intention rules* IR_1 *and* IR_2 *are applicable, and* $\mathsf{I}_{B3} = \{carry\}$*;*

Example 11. *Consider finally that* A^r *is in the situation of (Fig. 3(b)). The perception of* A^r *will be* $\Phi_4 = \{ball, freeTeammate, theirGoalieAway\}$*, and we can build the following arguments:*
$\mathcal{A}_8 : \{shoot \prec theirGoalieAway\}$,
$\mathcal{A}_9 : \{pass \prec freeTeammate\}$,
$\mathcal{A}_{10} : \{({\sim}carry \prec shoot), (shoot \prec theirGoalieAway)\}$.
Hence, $pass$*,* $shoot$ *and* ${\sim}carry$ *are warranted, and:*

- *for a cautious agent,* $\mathsf{D}^c_{C4} = \{shoot\}$*, intention rule* IR_3 *is applicable, and* $\mathsf{I}_{C4} = \{shoot\}$*;*
- *for a bold agent,* $\mathsf{D}^c_{B4} = \{shoot\}$*, intention rules* IR_3 *is applicable, and* $\mathsf{I}_{B4} = \{shoot\}$.

7 Application Example: Security System

In this section, we present an example consisting of a security-system agent. The system will be simplified in order to keep it easy to understand.

The security-system agent senses rooms of a building from four different sources: temperature, smoke, motion sensors, and video cameras. Whenever a temperature or smoke sensor is on, the agent will have a reason to call the firemen; analogously, if a motion sensor or a camera tells that an intruder might have entered to a room, the police should be called. These are not strict rules, but defeasible, as will be clear next (see Figure 4). The idea behind this setting is to have pairs of sensors acting as mutual backup, that is, we have smoke sensors as the backup for temperature sensors (and *vice versa*), and motion sensors as the backup for cameras (and *vice versa*).

Although sensor pairs provide robustness, they also bring about a few shortcomings, *e.g.*, a motion sensor in a room might detect that something is moving, while the corresponding camera is not showing any change in the image. Images coming from a camera may remain static for several reasons: perhaps someone attached a photograph to it, or the device could be simply malfunctioning. An analogous situation occurs when a room is signaled as having smoke but the temperature sensor placed there shows no activity. This generally means that one of the two sensors is not working properly. The filtering rules modelling abnormal situations like these are shown in Figure 4.

Abnormal situations within a room are handled by the agent, who will send a guard to that room (rules in Π_F, Figure 4). If the guard confirms that an intruder has entered to the room or that the room is on fire, it will manually trigger the corresponding alarm, providing a reason to the agent for calling the firemen or the police (last rule in Figure 4). Thus, once an alarm is fired, it will stop when a call (either to the police or firemen) is made, or when a guard arrives to a room and finds that everything is normal.

In this section, we will define an agent $A^s = \langle \mathsf{D}^s, (\Pi^s_{\mathsf{B}}, \Delta^s_{\mathsf{B}}), (\Pi^s_F, \Delta^s_F), \mathsf{T}^s, IR^s, p^s \rangle$. We start the description of the agent with its set of desires:

$$\mathsf{D}^s = \{send\_guard(R), call(firemen, R), call(police, R)\}$$

Desire $send\_guard(R)$ means that a guard could be sent to room R, and desires $call(firemen, R)$ and $call(police, R)$ give the possibility of calling the firemen and police because room R is on fire or an intruder entered to R, respectively.

In this case, there is no need to include negated literals in the set of desires, since, for instance, the security agent will never intend to fulfill the desire $\sim call(firemen, R)$. The system just will not make a call or send a guard to a room if there are no justified reasons to do it. This can be seen as just a design decision, but it turned to be a sensible representation.

A fundamental difference with the soccer domain examples is that the security-system agent does not require the selection of just one intention. This agent will select an arbitrary amount of intentions; it may even select no current desire as an intention (*i.e.*, $\mathsf{I}^s = \emptyset$). For instance, the agent could send several guards to certain rooms while making calls to both police and firemen regarding other rooms.

The security system will be managed by a cautious agent, *i.e.*, agent A^s will put in its set of current desires only the desires that are warranted from the DeLP-program $(\Pi^s_{\mathsf{B}} \circ \Pi^s_F, \Delta^s_{\mathsf{B}} \cup \Delta^s_F \cup \Delta^s_X)$. This choice will be justified at the end of this section.

The beliefs program $\mathcal{P}^s_{\mathsf{B}}$ has no rules, it will just consist of the set of perceived facts. The program (Π^s_F, Δ^s_F) of strict and defeasible filtering rules is shown in Figure 4.

$$\Pi_F^s = \left\{ \begin{array}{l} send\_guard(R) \leftarrow hi\_temp(R), {\sim}smoke(R) \\ send\_guard(R) \leftarrow {\sim}hi\_temp(R), smoke(R) \\ send\_guard(R) \leftarrow motion(R), {\sim}camera(R) \\ send\_guard(R) \leftarrow {\sim}motion(R), camera(R) \end{array} \right\}$$

$$\Delta_F^s = \left\{ \begin{array}{l} call(firemen, R) \mathrel{-\!\!\prec} hi\_temp(R) \\ call(firemen, R) \mathrel{-\!\!\prec} smoke(R) \\ {\sim}call(firemen, R) \mathrel{-\!\!\prec} hi\_temp(R), {\sim}smoke(R) \\ {\sim}call(firemen, R) \mathrel{-\!\!\prec} {\sim}hi\_temp(R), smoke(R) \\ call(police, R) \mathrel{-\!\!\prec} motion(R) \\ call(police, R) \mathrel{-\!\!\prec} camera(R) \\ {\sim}call(police, R) \mathrel{-\!\!\prec} motion(R), {\sim}camera(R) \\ {\sim}call(police, R) \mathrel{-\!\!\prec} {\sim}motion(R), camera(R) \\ call(Who, R) \mathrel{-\!\!\prec} manual\_alarm(Who, R) \end{array} \right\}$$

Fig. 4. Filtering rules for the security agent A^s

The set Π_F^s of strict rules models situations in which a couple of sensor differs and a guard has to be sent to a room, *e.g.*, when a camera detects no change, but the motion sensor placed in the same room says that something has moved.

Defeasible rules in Δ_F^s model reasons for and against making a call to police or firemen; for instance, if the temperature sensor signals heat in a certain room, the agent has a reason to call the firemen. However, if the corresponding smoke sensor has not fired, the agent will prefer not to make that call, but send a guard instead (modelled via strict rules). Regarding the last rule, if the manual alarm is fired by that guard, the call should be made immediately.

Note also that the filtering rules supporting calls to firemen or police refer to the room in which the danger was detected through variable R. This is important for the dialectical analysis to be performed over the same "situation". That is, if an argument for calling the police is posed and is under attack, it must be attacked by a counter-argument that speaks of the same room. This parameter also tells where firemen or police must go to.

The set IR^s of intention rules for the security agent are:

$$\begin{array}{l} IR_1 : \ (send\_guard(R) \Leftarrow \{\}, \{not\ manual\_alarm(W, R)\}) \\ IR_2 : \ (call(firemen, R) \Leftarrow \{\}, \{\}) \\ IR_3 : \ (call(police, R) \Leftarrow \{\}, \{\}) \end{array}$$

Observe that intention rule IR_1 has a constraint: a guard will not be sent to a room R in which a manual alarm has been triggered (variable W refers to whom should be called: police or firemen). This is because a guard is already there: the one who sounded the alarm. As will be clear below, this is best written as a constraint, rather than included into the filtering rules. Intention rules IR_2 and IR_3 specify that firemen and police should be called to go to room R whenever the head of the rule is a current desire.

It is important to note that, if the manual-alarm constraint of IR_1 is coded in the strict filtering rules, we should add this constraint in the body of each of the four rules. Keeping this constraint at intention-rules level allows us to write simpler filtering rules. For the agent A^s the policy p^s for selecting intentions will be simple, taking the set App of applicable intention rules and returning the set containing their heads:

$$p^s(App) = \{h \mid (h \Leftarrow P, C) \in App\}$$

Next, we introduce a series of sets of beliefs (B_0 through B_3) describing different scenarios. For each of them, the set of selected intentions will be calculated. In order to keep the example small and simple, we place our security agent in a building with two rooms: $r1$ and $r2$. The initial set of beliefs is:

$$\mathsf{B}_0 = \left\{ \begin{array}{ll} hi\_temp(r1), & smoke(r1), \\ {\sim}camera(r1), & {\sim}motion(r1), \\ {\sim}hi\_temp(r2), & {\sim}smoke(r2), \\ {\sim}camera(r2), & {\sim}motion(r2) \end{array} \right\}$$

Where positive literals represent a sensor that has fired, whereas negative literals mean the opposite. These beliefs, along with the filtering rules, give us two undefeated arguments for $call(firemen, r1)$:

$$\langle\{call(firemen, r1) \mathrel{-\!\!\prec} hi\_temp(r1)\}, call(firemen, r1)\rangle,$$
$$\langle\{call(firemen, r1) \mathrel{-\!\!\prec} smoke(r1)\}, call(firemen, r1)\rangle.$$

Then, the set of current desires is $\mathsf{D}^c_0 = \{call(firemen, r1)\}$, which means that the only applicable intention rule is IR_2, and the set of selected intentions consists of the head of IR_2, that is $\mathsf{I}^s_0 = \mathsf{D}^c_0 = \{call(firemen, r1)\}$.

Observation: To avoid the system to keep sending guards to a room, we will assume that queries about desires are performed only when the set of beliefs has changed.

Suppose now a different situation, in which not only the temperature and smoke sensors in room $r1$ had fired, but also did the motion sensor in room $r2$. The new set of beliefs is:

$$\mathsf{B}_1 = \left\{ \begin{array}{ll} hi\_temp(r1), & smoke(r1), \\ {\sim}camera(r1), & {\sim}motion(r1), \\ {\sim}hi\_temp(r2), & {\sim}smoke(r2), \\ {\sim}camera(r2), & motion(r2) \end{array} \right\}$$

As before, $call(firemen, r1)$ has two undefeated arguments. In addition, now there is one argument for calling the police:

$$\langle\{call(police, r2) \mathrel{-\!\!\prec} motion(r2)\}, call(police, r2)\rangle,$$

which is attacked by:

$$\langle\{{\sim}call(police, r2) \mathrel{-\!\!\prec} motion(r2), {\sim}camera(r2)\}, {\sim}call(police, r2)\rangle.$$

Since the argument for not calling the police is more specific than the other, the argument supporting $call(police, r2)$ is defeated and does not belong to D^c_1 (note that $\sim call(police, r2)$ is warranted). In addition to this, there is an empty argument for $send\_guard(r2)$ from the strict rule $(send\_guard(R) \leftarrow motion(R), \sim camera(R))$. Thus, we have $\mathsf{D}^c_1 = \{call(firemen, r1), send\_guard(r2)\}$. Intention rule IR_1 is applicable, because its precondition holds: $send\_guard(r2) \in \mathsf{D}^c_1$, and its constraint is satisfied: $manual\_alarm(W, r2) \notin \mathsf{B}_1$. Intention rule IR_2 is also applicable, because $call(firemen, r1) \in \mathsf{D}^c_1$. Hence, again we have that the current set of selected intentions equals the set of current desires, *i.e.*, $\mathsf{I}^s_1 = \mathsf{D}^c_1$. This will happen whenever the manual alarm is not triggered, since the set of intention rules IR^s is quite simple (rules have no preconditions nor constraints, excepting IR_1), and so is the policy (to take the head of every applicable intention rule as a selected intention).

Now suppose that the situation in room $r1$ is now normal, but the motion sensor in room $r2$ fired, and a guard has been sent to that room to check if an intruder has effectively entered there. Let us assume that the guard finds a thief in room $r2$. Then, the guard triggers the manual alarm, which changes the set of beliefs of the security-system agent:

$$\mathsf{B}_2 = \left\{ \begin{array}{ll} \sim hi\_temp(r1), & \sim smoke(r1), \\ \sim camera(r1), & \sim motion(r1), \\ \sim hi\_temp(r2), & \sim smoke(r2), \\ \sim camera(r2), & motion(r2), \\ manual\_alarm(police, r2) & \end{array} \right\}$$

The following arguments for and against $call(police, r2)$ are built:

$$\begin{aligned} \mathcal{A}^s_1 &= \langle \{call(police, r2) \prec motion(r2)\}, call(police, r2)\rangle, \\ \mathcal{A}^s_2 &= \langle \{\sim call(police, r2) \prec motion(r2), \sim camera(r2)\}, \sim call(police, r2)\rangle, \\ \mathcal{A}^s_3 &= \langle \{call(police, r2) \prec manual\_alarm(police, r2)\}, call(police, r2)\rangle. \end{aligned}$$

Argument $\mathcal{A}^s_1$ has $\mathcal{A}^s_2$ as proper defeater, since the latter is more specific than the former. In turn, $\mathcal{A}^s_2$ is blocked by defeater $\mathcal{A}^s_3$, reinstating $\mathcal{A}^s_1$. Thus, the desire $call(police, r2)$ is now warranted, and belongs to D^c_2. Desire $send\_guard(r2)$ is also in D^c_2 from the strict rule $(send\_guard(R) \leftarrow motion(R), \sim camera(R))$. Therefore, $\mathsf{D}^c_2 = \{call(police, r2), send\_guard(r2)\}$. Note that now intention rule IR_1 is not applicable, because its constraint $(not\ manual\_alarm(police, r2))$ does not hold. Here, the only applicable intention rule is IR_3. Thus, the set of selected intentions is $\mathsf{I}^s_2 = \{call(police, r2)\}$, which differs from D^c_2. It is a sensible decision not to send a guard to a room when the police is already being sent there by a guard in that room.

Finally, consider that sensors do not detect anything abnormal, then:

$$\mathsf{B}_3 = \left\{ \begin{array}{ll} \sim hi\_temp(r1), & \sim smoke(r1), \\ \sim camera(r1), & \sim motion(r1), \\ \sim hi\_temp(r2), & \sim smoke(r2), \\ \sim camera(r2), & \sim motion(r2), \end{array} \right\}$$

This set of beliefs builds no arguments for any desire from the filtering rules. Then, the set of current desires is empty, and so is the set of selected intentions, *i.e.*, $\mathsf{D}^c_3 = \mathsf{I}^s_3 = \emptyset$. The system will remain in this state until a sensor is fired.

The choice of a cautious agent instead of a bold one is clear when analyzing the latter case. A bold agent would select every desire as a current desire, since there are no arguments for nor against any of them:

$$\mathsf{D}_3'^c = \left\{ \begin{array}{l} send\_guard(r1), send\_guard(r2), \\ call(firemen, r1), call(firemen, r2), \\ call(police, r1), call(police, r2) \end{array} \right\}$$

Regarding intention rules, all of them will be applicable, and therefore $\mathsf{I}_3'^s = \mathsf{D}_3'^c$. This means that guards will be sent to rooms $r1$ and $r2$, and both the police and firemen will be called to check on both rooms. Clearly, this is not the intended behavior for the security-system agent.

8 Related Work

The use of defeasible argumentation in BDI architectures is not new and it was originally suggested in [7], and more recently in [8]. Also in [9] and [10] a formalism for reasoning about beliefs and desires is given, but they do not use argumentation.

Recently, Rahwan and Amgoud [11] have proposed an argumentation-based approach for practical reasoning that extends [12] and [13], introducing three different instantiations of Dung's framework to reason about beliefs, desires and plans, respectively. This work is, in our view, the one most related to ours. Both approaches use defeasible argumentation for reasoning about beliefs and desires (in their work, they also reason about plans, but this is out of the scope of our presentation). Like us, they separate in the language those rules for reasoning about belief from those rules for reasoning about desires; and, in both approaches, it is possible to represent contradictory information about beliefs and desires. Both approaches construct arguments supporting competing desires, and they are compared and evaluated to decide which one prevails. Their notion of *desire rule* is similar to our *filtering rules*.

In their approach, two different argumentation frameworks are needed to reason about desires: one framework for beliefs rules and another framework for desires rules. The last one depends directly on the first one, and since there are two kinds of arguments, a policy for comparing mixed arguments is given. In our case, only one argumentation formalism is used for reasoning with both types of rules. In their object language, beliefs and desires include a certainty factor for every formula, and no explicit mention of perceived information is given. In our case, uncertainty is represented by defeasible rules [2] and perceived beliefs are explicitly treated by the model. Besides, the argumentation formalism used in their approach differs from ours: their comparison of arguments relies on the certainty factor given to each formula, and they do not distinguish between proper and blocking defeaters. Another fundamental difference is that we permit the definition of different types of agents. This feature adds great flexibility in the construction of an agent.

9 Conclusions

We have shown how a deliberative agent can represent its perception and beliefs using a defeasible logic program. The information perceived directly from the environment is represented with a subset of perceived beliefs that is dynamically updated, and a set formed with strict rules and facts represent other static knowledge of the agent. In addition to this, defeasible argumentation is used to warrant agents (derived) beliefs. Strict and defeasible filtering rules have been introduced to represent knowledge regarding desires. Defeasible argumentation is used for selecting a proper desire that fits in the particular situation the agent is involved. With this formalism, agents can reason about its desires and select the appropriate ones.

We allow the representation of different agent types, each of which will specify a different way to perform the filtering process. In our approach, an intention is a current desire that the agent can commit to pursue. The agent is provided with a set of intention rules that specify under what conditions an intention could be achieved. If there is more than one applicable intention rule, then a policy is used to define a preference criterion among them. Thus, intention policies give the agent a mechanism for deciding which intentions should be selected in the current situation.

In this work, we have shown how to implement two rather different kinds of agents using our model. We discussed their similarities and differences, stressing the point of the selection of the set of intentions, which is bound to be a singleton in one application, whereas is unrestricted in the other. Another difference regards to the way each of these agents perceive and gather beliefs. Regarding the sets of desires of both applications, they do not have any important structural difference; in fact, they coincide in not having complementary literals. However, it is difficult to conceive an application domain with a set of desires containing complementary literals. Usually, an argument concluding the complement of a desire has the purpose of "stopping" the justification (in the sense of warrant) of that desire, rather than supporting the opposite desire.

As future work, further research will be directed towards the improvement of the implementation of the proposed architecture. We plan to use a DeLP-Server [14], which provides a Defeasible Logic Programming reasoning service and allows client-agents to perform contextual queries.

References

1. Rotstein, N., García, A., Simari, G.: Reasoning from desires to intentions: A dialectical framework. In: Proceedings of the 22nd. AAAI Conference on Artificial Intelligence, pp. 136–141 (2007)
2. García, A., Simari, G.: Defeasible logic programming: An argumentative approach. Theory Practice of Logic Programming 4(1), 95–138 (2004)
3. Lifschitz, V.: Foundations of logic programming. In: Brewka, G. (ed.) Principles of Knowledge Representation. CSLI, pp. 69–127 (1996)
4. Rotstein, N., García, A.: Defeasible reasoning about beliefs and desires. In: Proc. of the 11th NMR, pp. 429–436 (2006)
5. Falappa, M., Kern-Isberner, G., Simari, G.: Belief revision, explanations and defeasible reasoning. Artificial Intelligence Journal 141, 1–28 (2002)
6. Fuhrmann, A.: An Essay on Contraction. In: Studies in Logic, Language and Information, CSLI Publications, Stanford, CA (1997)
7. Bratman, M.E., Israel, D., Pollack, M.: Plans and resource-bounded practical reasoning. In: Cummins, R., Pollock, J.L. (eds.) Philosophy and AI: Essays at the Interface, pp. 1–22. MIT Press, Cambridge (1991)

8. Parsons, S., Sierra, C., Jennings, N.: Agents that reason and negotiate by arguing. Journal of Logic and Computation 8(3), 261–292 (1998)
9. Thomason, R.: Desires and defaults: A framework for planning with inferred goals. In: Proc. of the seventh KR, pp. 702–713 (2000)
10. Broersen, J., Dastani, M., Hulstijn, J., Huang, Z., van der Torre, L.: The boid architecture: conficts between beliefs, obligations, intentions and desires. In: Proc. of 5th Int. Conf. on Autonomous Agents, pp. 9–16. ACM Press, New York (2001)
11. Rahwan, I., Amgoud, L.: An argumentation-based approach for practical reasoning. In: Proc. of the 5th AAMAS (2006)
12. Amgoud, L.: A formal framework for handling conflicting desires. In: Nielsen, T.D., Zhang, N.L. (eds.) ECSQARU 2003. LNCS (LNAI), vol. 2711, pp. 552–563. Springer, Heidelberg (2003)
13. Amgoud, L., Cayrol, C.: A reasoning model based on the production of acceptable arguments. Annals of Mathematics and Artificial Intelligence 34(1-3), 197–215 (2002)
14. García, A., Rotstein, N., Tucat, M., Simari, G.: An Argumentative Reasoning Service for Deliberative Agents. In: Zhang, Z., Siekmann, J.H. (eds.) KSEM 2007. LNCS (LNAI), vol. 4798, Springer, Heidelberg (2007)