A Framework for Multi-criteria Argumentation-Based Decision Making within a BDI Agent

Cecilia Sosa Toranzo, Marcelo Errecalde and Edgardo Ferretti
Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)
Universidad Nacional de San Luis
Ejército de los Andes 950 - (D5700HHW) San Luis - Argentina
e-mails:{csosatoranzo, merreca, ferretti}@unsl.edu.ar

ABSTRACT

The BDI model, as a practical reasoning architecture aims at making decisions about what to do based on cognitives notions as beliefs, desires and intentions. However, during the decision making process, BDI agents also have to make background decisions like choosing what intention to achieve next from a set of possibly conflicting desires; which plan to execute from among the plans that satisfy a given intention; and whether is necessary or not to reconsider current intentions. With this aim, in this work, we present an abstract framework which integrates a *Possibilistic Defeasible Logic Programming* approach to decision making in the inner decision processes within BDI agents

Keywords: Agreement Technologies, Multi-criteria Decision Making, BDI, Argumentation, Possibilistic Defeasible Logic Programming.

1. INTRODUCTION

The BDI model is a particular decision making model based on cognitive notions, namely: Belief, Desires and Intentions. This model is very relevant because of its similarity with human reasoning which makes it easy to understand it, the theoretical underpinning it has [1, 2, 3], as well as its applicability to solve real-world problems [4, 5, 6, 7].

BDI architecture is inspired from Bratman's work on *practical reasoning* [2]. Practical reasoning (PR) aims at deciding what to do in a given situation and thus is directed towards action. However, besides deciding which action perform next, BDI agents also have to decide: (a) which intention to achieve from a set of possibly conflicting desires, (b) which plan to execute from among the plans that satisfy the chosen intention, and (c) whether is necessary or not to reconsider current intentions. That is, BDI model also implies making background decisions.

Some of the issues mentioned above have been tackled in previous works. Casali *et al.* [8] present a general framework to define graded BDI agent architectures, where degrees in BDI models are used to set different levels of preferences or rejections on desires and preferences at intentions level to model the cost/benefit trade-off of reaching a goal. In [9], ideas from argumentation are combined with desire and planning rules, to give a formal account on how consistent sets of intentions can be obtained from a conflicting set of desires. In this work, the issue of choosing one desire from among the conflicting ones is not tackled and those conflict-free or which become undefeated from their arguments, are chosen.

A general framework for practical reasoning based on an abstract argumentative machinery is presented in [10], where it is argued that PR is a three-step process involving two inference steps (generating desires to be accomplished and plans to achieve them) and one decision step. This last step uses a single decision criterion for selecting the intention to be pursued. To the best of our knowledge, at present, there are no proposals which clearly formulate how these choices are made in BDI agent's inner decision processes. In this way, the main goal of this paper aims at incorporating in a generic way, multi-criteria decision making in BDI agent's inner decision processes. In particular, an argumentation-based approach to multi-criteria decision making is used [11]. In this respect, some proposals exist [12, 13], aiming at incorporating argumentation-based approaches within BDI agents.

In [12], a framework where defeasible argumentation [14] is used for reasoning about beliefs, desires and intentions, is defined. A dialectical filtering process is introduced to obtain a subset of the agent's desires containing only those that are achievable in the current situation. The agent is provided with a set of intention rules that specifies under what conditions an intention could be achieved. When more than one intention is possible, a policy will be used to choose among them. Moreover, [13] designs and implements a travel assistant agent by using argumentation in the main process of a generic framework. This framework integrates argumentation-based inference and web services technologies into the design of a BDI system.

The rest of the paper is organized as follows. Section 2 briefly introduces the BDI model to provide

the background concepts underlying the proposed abstract framework (Section 3), which integrates multi-criteria argumentation-based decision making (Section 4) in the inner decision processes of the BDI architecture. Then, this framework is exemplified in the TILEWORLD domain (Section 5). Finally, Section 6 draws the conclusions and briefly describes possible future work.

2. BDI MODEL

Belief-Desires-Intentions models (BDI) have been inspired from the philosophical tradition on understanding *practical reasoning*. This kind of reasoning can be conceived as the process of deciding what action perform next to accomplish a certain goal. Practical reasoning involves two important processes, namely: deciding *what* states of the world to achieve and *how* to do it. The first process is known as *deliberation* and its result is a set of intentions. The second process, so-called *means-ends reasoning*, involves generating actions sequences to achieve intentions.

The mental attitudes of a BDI agent on its beliefs, desires and intentions, represent its informational state, motivational state and decision state, respectively. The BDI architecture defines its cognitive notions as follows:

BELIEFS: Partial knowledge the agent has about the world.

DESIRES: The states of the world that the agent would ideally like to achieve.

INTENTIONS: Desires (states of the world) that the agent has committed (dedicated resources) to achieve.

These cognitive notions are implemented as data structures in the BDI architecture, which also has an interpreter that manipulates them to select the most appropriate actions to be performed by the agent. This interpreter performs the deliberation and means-ends reasoning processes aforementioned, and it is shown in Algorithm 1 (as proposed in [15]). It assumes that an explicit representation of desires (D), belief (B) and intentions (I) exist, within the agent. Besides, with Bel, Des and Int we will denote the sets of all the beliefs, desires and intentions the agent will possibly have, respectively. The agent's perceptions will be represented by ρ and similarly, Per will denote the set of all possible perceptions. Regarding plans, they will be referred as "recipes" to achieve intentions. Therefore, π will be used to denote plans and $Plan$ will designate the set of all the plans (on some set of actions Ac).

Lines 1−3 in Algorithm 1 initialize beliefs, intentions and the plan. The main control loop comprises lines 4−19. In lines 5−6, the agent perceives and updates its beliefs while in line 7 it decides whether to reconsider or not. In lines 8−12 it deliberates and in line 11, it generates a plan to achieve its intentions. Lines 14−17 show that an action of the current plan is executed. Because the purpose of the functions in this loop can be easily derived from their names, due to space constraints we omit their formalizations but the interested reader should refer to [15].

A usual problem in designing practical reasoning agents lies in getting a good balance among deliberation, means-ends reasoning and actions execution. It is clear that, in some point of time, an agent should drop some of its intentions, because they were already achieved, they are impossible to be achieved or makes no sense to do it, etc. Likewise, when opportunities arise to achieve new desires, the agent should generate intentions to accomplish them. Thus, as mentioned above it is important for an agent to *reconsider its intentions*. However, intentions reconsideration is costly in terms of time and computational resources. It can happen that some of the actions from the executing plan might fail in achieving the intended results, hence *replanning* capabilities should be provided. Both replanning and intentions reconsideration (if performed) must be carried out during the execution phase of the chosen actions.

Algorithm 1 BDI Agent control loop

```
1:  B ← B_0
2:  I ← I_0
3:  π ← null
4:  while true do
5:      get next percept ρ
6:      update B on the basis of ρ
7:      if reconsider(B,I) then
8:          D ← options(B,I)
9:          I ← filter(B,D,I)
10:         if not sound(π,I,B) then
11:             π ← plan(B,I)
12:         end if
13:     end if
14:     if π ≠ ∅ then
15:         α ← hd(π)
16:         execute(α)
17:         π ← tail(π)
18:     end if
19: end while
```

3. INTEGRATION FRAMEWORK

As mentioned above, the BDI model uses the cognitive notions of beliefs, desires and intentions to decide what to do, but also, inner decisions exist related to these high-level decisions which, in our view, have not been clearly detailed in previous works. That is why, in this section we propose an abstract framework which integrates multi-criteria decision making to solve inner decision making in a BDI agent.

In Section 2 was referred that a BDI agent comprises two fundamental processes, namely, deliberation and means-ends reasoning, which are followed

by a plan execution stage. Within these processes (deliberation, means-ends reasoning and execution) the following inner decisions can be made:

CHOICE AMONG CONFLICTING DESIRES: *deliberation* requires to commit to an intention from among conflicting desires.

CHOICE BETWEEN PLANS: during *means-ends reasoning* it might be necessary to choose from among plans which achieve the same intention, that is, deciding which action perform to achieve a particular intention.

INTENTIONS RECONSIDERATION: during the *execution* process (of only one plan or a mega-plan involving all the plans the agent has committed to) decisions should be made with respect to whether reconsider or not current intentions based on the dynamics of the environment, and if so, if new intentions should be adopted or current intentions should be dropped.

All in all, our BDI architecture will incorporate an *Inner Decision Making Component (IDMC)* which will make inner decisions with respect to the different alternatives and the multiple criteria provided to the agent. In our proposal, to select the best alternative from a given set of alternatives, the agent will have the $select(\cdot,\cdot,\cdot)$ function that will return the choice made by IDMC. This function will be used (within this framework) in all the inner decision processes a BDI agent has. It will receive as input parameters: (1) a set A of candidate alternatives, (2) the set C containing the criteria that will be used to compare alternatives among each other, and (3) the preferences $\mathcal{P}$, composed by a preference order among criteria and a preference order among the possible values an alternative can take for each particular criterion.

In the following subsections, there will be described in more detail those functions called in the BDI interpreter, which use $select(\cdot,\cdot,\cdot)$ function, to make the inner decisions mentioned at the beginning of this section.

3.1. Deliberation

Based on Algorithm 1, deliberation process can be considered as composed by two functions:

- $options(\cdot,\cdot)$: which returns a set of *acceptable options* (desires) considering the agent's belief.
- $filter(\cdot,\cdot,\cdot)$: which returns the set of alternatives the agent has committed to.

Once desires have been obtained by using $options(\cdot,\cdot)$ function, the agent may find conflicting options since desires set might not be consistent. Hence, the agent must *choose* one alternative among the competing ones to commit to. Then, $filter(\cdot,\cdot,\cdot)$ function will accept those non-conflicting options and from among the conflicting ones, only one will be selected. Next, the agent will commit to the surviving options (intentions) of the filtering process. In (1) the formal definition of $filter(\cdot,\cdot,\cdot)$ function is given. To select one of the conflicting desires, the agent will call $select(\cdot,\cdot,\cdot)$ within $filter(\cdot,\cdot,\cdot)$ function, whose generic algorithm is shown in Algorithm 2. It is worth noting that in simplest cases, where all the alternatives are conflicting among each other (see example in Section 5), $filter(\cdot,\cdot,\cdot)$ function is directly conceived as the $select(\cdot,\cdot,\cdot)$ function.

$$filter : \wp(Bel) \times \wp(Des) \times \wp(Int) \rightarrow \wp(Int) \quad (1)$$

Algorithm 2 Filtering of alternatives

function: *filter*(beliefs B, desires D, intentions I) **return** I

```
1:  C ← selection criteria
2:  P ← user's preferences
3:  D' ← D
4:  while D' ≠ ∅ do
5:      d ← remove-any(D')
6:      if feasible(d,B) then
7:          for all d' ∈ D do
8:              if competing(d,d') then
9:                  add(d',A)
10:                 remove(d', D')
11:             end if
12:         end for
13:         if A = ∅ then
14:             add(d,I)
15:         else
16:             add(d,A)
17:             a_I ← select(A,C,P)
18:             add(a_I,I)
19:         end if
20:     end if
21: end while
22: return I
```

3.2. Means-ends Reasoning

Means-ends reasoning is the process aiming to decide how to achieve an *end* (*i.e.,* an intention) by means of the available actions the agent has. In Artificial Intelligence community, means-ends reasoning is best known as *planning*. A planning algorithm outputs a "plan", that is, the sequence of actions to be performed and which was previously referred in this paper as a "recipe".

In the BDI interpreter depicted in Algorithm 1, means-ends reasoning is achieved by calling $plan(\cdot,\cdot)$ function. This function, based on beliefs and current intentions together with the actions available to the agent, determines a plan to achieve the intentions. On the grounds that several plans may exist to achieve a certain intention, a choice among them

should be made considering issues like: cost execution (physical resources needed), time execution, sensitivity to changes in the environment, etc. The implementation of $plan(\cdot,\cdot)$ function based on $select(\cdot,\cdot,\cdot)$ function, is presented in Algorithm 3.

Algorithm 3 Planning

function: *plan*(beliefs B, intentions I) **return** Π

```
1: C ← selection criteria
2: 𝔓 ← user's preferences
3: C′ ← criteria for sort plans
4: for i ∈ I do
5:    P ← find-plans(B,i)
6:    if singleton(P) then
7:       add(π ∈ P,Π)
8:    else
9:       π ← select(P,C,P)
10:      add(π,Π)
11:   end if
12:   sort(Π,C′)
13: end for
14: return Π
```

3.3. Execution

One design issue in BDI agents concerns defining the intention reconsideration policy [15, 16, 17]. This policy will define under which circumstances the BDI agent will use computational resources to deliberate about its intentions. At present there is no consensus on when or how an agent should reconsider its intentions. Current proposals consider the agents' *commitment levels*, which range from *cautious* agents (which reconsider their intentions after each action execution) to *bold* agents (that perform no reconsideration until the current plan has been completely executed).

Kinny and Georgeff investigated the efficiency of these policies in different kind of environments [16] but the intention reconsideration policy is defined in the design stage of the agent, which makes impossible to modify this policy in execution time. It is evident that this is not a practical solution for agents operating in dynamic and changing environments. In this respect, Schut and Wooldridge [17] proposed a framework that allows the agent choosing by itself what policy to follow based on the current state of the world. The main idea underlying this work is that an intention reconsideration policy can be conceived as a meta-level control process which selects whether to deliberate or act. This proposal is based on the *discrete deliberation scheduling* framework [18], where deliberations are treated as if they were actions.

In [17], the proposed model incorporates decision making in the intention reconsideration process of a BDI agent. To determine the best possible action, the maximum expected utility is considered which means considering only one criterion to solve the decision problem. In this way, our work extends [17] to apply multi-criteria decision making when choosing between acting or deliberating.[1]

Integrating the BDI model with the discrete deliberation scheduling framework, from a multi-criteria point of view, involves implementing $reconsider(\cdot,\cdot)$ function. In this way, a distinction should be made between external actions and inner actions. External actions, Ac_{ext}, change the environment where the agent is located, whereas inner actions, Ac_{int}, affect the agent's internal state. It holds that $Ac = Ac_{ext} \cup Ac_{int}$ and it is assumed that $Ac_{ext} \cap Ac_{int} = \emptyset$. As it can be observed in Algorithm 4, in this model, the agent must choose between a default action $a_{def} \in Ac_{ext}$ (acting) or an inner action $a_{del} \in Ac_{int}$ (deliberating). Let $\pi \in Plan$ be a plan composed by actions $\pi[0], \ldots, \pi[n-1]$, where $\pi[i] \in A_{ext}$ and n denotes the plan length; in any execution time it holds that a_{def} is $\pi[0]$.

The use of the BDI model makes the treatment of deliberation at a very abstract level, since deliberation is considered as a way of changing the intentions set and is referred as a simple inner action. When $reconsider(\cdot,\cdot)$ function returns true, this means that deliberating was decided, while if returned false, this means that acting was chosen. To decide between both meta-actions, the agent should be provided well-established comparison criteria, then it has to compute the actions values considering their consequences and it also has to estimate the benefit of deliberating for a_{del}.

Algorithm 4 Intention Reconsideration

function: *reconsider*(beliefs B, intentions I) **return** bool

```
1: C ← selection criteria
2: P ← user's preferences
3: get current plan π from I
4: if π = ∅ then
5:    return true
6: end if
7: a_def ← π[0]
8: A ← {a_def, a_del}
9: a_selec ← select(A, C, P)
10: if a_selec = a_del then
11:    return true
12: end if
13: return false
```

In the following section, a concrete argumentation-based approach is proposed to instantiate the abstract framework presented in this section. The argumentation-based mechanism will be implemented by the IDMC, which will return to the BDI

[1] In this case, "deliberating" refers to the deliberative and planning stages comprised in the BDI interpreter.

interpreter the choice made, as result of the inner decision making process.

4. THE ARGUMENTATION-BASED INSTANTIATION

The argumentation-based decision framework described in this section is formally related to the *choice-based approach* (CBA) to decision making, as stated in [11]. The CBA takes as primitive object the choice behaviour of the individual, which is represented by means of a *choice structure* $(\mathcal{B}, \mathbf{C}(\cdot))$. $\mathcal{B}$ is a set of subsets of X (the set containing all the available alternatives to the decision maker). Each set $B \in \mathcal{B}$, represents a set of alternatives (or *choice experiment*) that can be conceivably posed to the decision maker. $\mathbf{C}(\cdot)$ is a *choice rule* which basically assigns to each set of alternatives $B \in \mathcal{B}$ a non-empty set that represents the alternatives that the decision maker *might* choose when presented the alternatives in B ($\mathbf{C}(B) \subseteq B$ for every $B \in \mathcal{B}$). When $\mathbf{C}(B)$ contains a single element, this element represents the *individual's choice* among the alternatives in B. The set $\mathbf{C}(B)$ might, however, contain more than one element and in this case they would represent the *acceptable alternatives* in B for the decision maker.

This decision framework is conceptually composed by three components. The first component is set X. The second component, the epistemic component, represents the agent's knowledge and preferences, and the third one is the decision component. Formally, the argumentation-based decision framework is a triple $\langle X, \mathcal{K}, \Gamma \rangle$ where:

X is the set of all the *possible alternatives* that can be presented to the decision maker.

$\mathcal{K}$ is the *epistemic component* of the decision maker (see Definition 4.5 from [11]). Formally, $\mathcal{K}$ is a 5-tuple, $\mathcal{K} = \langle \mathcal{C}, >_{\mathcal{C}}, ACC, \Pi, \Delta \rangle$ where:

- $\mathcal{C}$ is a set of *comparison literals* representing the preference criteria that the decision maker will use to compare the elements in X. Let $C = \{C_1, \ldots, C_n\}$ $(n > 0)$ be the set of preference criteria that will be used to compare the elements in X, each criterion C_i has associated a *comparision literal* $c_i(\cdot, \cdot)$ that states the preference between two alternatives of X, based on their attribute values. Then, $\mathcal{C} = \{c_1(\cdot, \cdot), \ldots, c_n(\cdot, \cdot)\}$.
- $>_{\mathcal{C}}$ is a strict total order over the elements of $\mathcal{C}$. (Definition 4.2 from [11]).
- ACC is a user-specified *aggregation function* that aggregate necessity degrees. ACC must satisfy specific properties (see [11]).
- Π is a set of *certain clauses*.
- Δ is a set of *uncertain clauses*.

  $P(\Pi, \Delta)$ is a conformant P-DeLP program (see Definition 4.3 from [11]).

Γ is the *decision component*. It is a set with two decision rules:[2]

$DR1 : \{W\} \overset{B}{\Leftarrow} \{bt(W,Y)\}, \; not\{bt(Z,W)\}\rangle$
$DR2 : \{W,Y\} \overset{B}{\Leftarrow} \{sp(W,Y)\}, \; not\{bt(Z,W)\}\rangle$
with $B \subseteq X$.

Rule DR1 states that an alternative $W \in B$ will be chosen, if W is better than another alternative Y and there is not a better alternative Z than W. Besides, rule DR2 says that two alternatives $W, Y \in B$ with the same properties will be chosen if there is not a better alternative Z than W and Y.

Let $B \in \mathcal{B}$ be a set of alternatives posed to the agent and $\langle X, \mathcal{K}, \Gamma \rangle$ be the agent's decision framework. Let $\{D_i \overset{B}{\Leftarrow} P_i, not\ T_i\}_{i=1\ldots n} \subseteq \Gamma$ be the set of applicable decision rules with respect to $\mathcal{K}$. The set of *acceptable alternatives* of the agent will be $\Omega_B = \bigcup_{i=1}^{n} D_i$. The set Ω_B is a subset of B and if Ω_B contains a single element, that element is the decision maker's individual choice from among the alternatives in B. However, if Ω_B contains more than one element, then they represent acceptable alternatives that the agent might choose.

In Algorithm 5, a general algorithm which implements a choice rule $\mathbf{C}(\cdot)$ is presented. As it can be observed function μ has as input parameter a choice experiment (B). A choice experiment is a set containing at least one element, hence, this function returns *failure* if receives as argument an empty set (step 1). If the choice experiment has one element, then it is thus returned as solution since there is only one trivial choice to be made (step 2). Then, if a non-empty set was received as parameter, the resulting set sol is initialized (step 3) and a local copy (ch) of the original choice experiment is made (step 4). The computing process to determine the set of acceptable alternatives ends when ch becomes empty (step 6), thus exiting the main loop (step 5) returning the computed set of acceptable alternatives sol (step 13). While there are alternatives in ch, an alternative is removed from this set and is assigned to h (step 7). If there is not a better alternative than h in the choice experiment (step 9) and h is better than any other alternative in the choice experiment (step 8), then h is added to the resulting set sol (step 10), otherwise is discarded (step 9). Besides, if h is not better than any other alternative in the choice experiment (step 8), but there is no other alternative (let us denoted it as h') in the choice experiment better than h (step 11), then it holds that h and h' have the same properties, and they are the best, therefore h is added to the resulting set sol (step 12).

[2]Due to space constraints, the literals $better(\cdot, \cdot)$ and $same\_prop(\cdot, \cdot)$ in [11], will be referred as $bt(\cdot, \cdot)$ and $sp(\cdot, \cdot)$, respectively.

It is worth mentioning, that in turn (when selected in step 7) h' will also be added to sol.

Based on the above-mentioned framework, function $select(\cdot,\cdot,\cdot)$ executes the steps shown in Algorithm 6 to choose from among the alternatives in B.

Algorithm 5 Compute Acceptable Alternatives

function μ(choice-experiment) **returns** non-empty-set-of-alternatives, or failure
1: **if** EMPTY?(choice-experiment) **then return** failure
2: **if** SINGLETON?(choice-experiment) **then return** choice-experiment
3: $sol \leftarrow \emptyset$
4: $ch \leftarrow$ choice-experiment
5: **loop do**
6: **if** EMPTY?(ch) **then exit**
7: $h \leftarrow$ REMOVE-ELEMENT(ch)
8: **if** IS-h-BETTER-THAN-ANY-OTHER?(choice-experiment) **then**
9: **if** ANY-BETTER-THAN-h?(choice-experiment) **then** discard h
10: **else** ADD-ELEMENT(sol,h)
else
11: **if** ANY-BETTER-THAN-h?(choice-experiment) **then** discard h
12: **else** ADD-ELEMENT(sol,h)
13: **return** sol

Algorithm 6 Computation for alternatives selection

function *select*(alternatives B, criteria C, preferences $\mathcal{P}$) **returns** non-empty-set-of-alternatives, or failure

1: Define the comparison literal for each $C_i \in C$
2: Define $>_{\mathcal{C}}$ according to the preferences of each criterion in $\mathcal{P}$
3: Build a conformant program $P(\Pi, \Delta)$ (as defined in [11])
4: **return** Evaluation of function $\mu(B)$

5. EXAMPLE: THE TILEWORLD

The TILEWORLD experimental domain [19] is a grid environment containing agents, tiles, holes and obstacles. The agent's objective consists of scoring as many points as possible by pushing the tiles into the holes to fill them in. The agent is able to move up, down, left, or right, one cell at a time, having as only restriction that obstacles must be avoided. This environment is dynamic, so that holes and tiles may randomly appear and disappear in accordance to a series of world parameters, which can be varied by the experimenter.

A BDI agent for the TILEWORLD can be implemented as follows: the agent's beliefs consist of its perceptions about the objects locations, as well as the score and time-out time for all the holes. Desires are the holes to be filled in, and the current intention (IH) aims at filling a particular hole right now. The means-end reasoner basically is a special-purpose route planner, which guides the agent to a particular tile that must be pushed into the hole to be filled in. Figure 1 shows a hypothetical scene in which the framework proposed in Section 3 will be used.

The agent gets its perception and updates its beliefs, in order to deliberate about what intention to achieve next. During deliberation it gets its reachable holes (options), *i.e.,* those which are not surrounded by obstacles and their time-out times are higher or equal to the distances from the agent to the holes. Then, as described in Section 5.1, filtering stage takes place where one of the reachable holes is selected and becomes the intention to be achieved (IH).

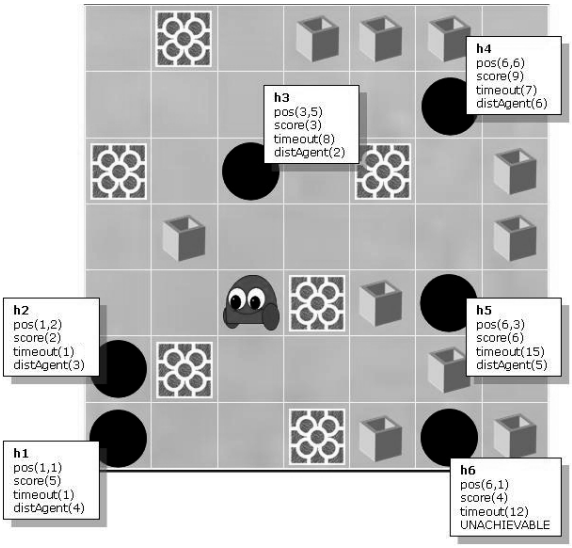

Figure 1: Tileworld scene

5.1. Filtering

In this case, all options are conflicting each other, since it is not possible to fill in more than one hole at a time. Hence, all reachable holes will serve as input to $selec(\cdot,\cdot,\cdot)$ function. In this way, $B = \{h_3, h_4, h_5\}$, $C = \{C_1, C_2, C_3, C_4\}$ where: $C_1 = \text{score}$, $C_2 = \text{timeout}$, $C_3 = \text{distAgent}$, $C_4 = \text{tileAvail}$ (distance to the nearest tile) and

$$>_{\mathcal{C}} = \{(distAgent, timeout), (distAgent, tileAvail), (timeout, tileAvail), (score, distAgent), (score, timeout), (score, tileAvail)\}$$

Table 1 presents preferences for each criterion. Table 2 shows the alternatives and their respective values for each criterion, while Table 3 shows the alternatives and their respective values normalized to interval $[0, 1]$. Likewise, following the approach

from [11], a conformant P-DeLP program would be:

$$\Delta = \left\{ \begin{array}{l} (score(h_4, h_3), 0.92) \\ (score(h_4, h_5), 0.83) \\ (score(h_5, h_3), 0.83) \\ (distAgent(h_3, h_5), 0.62) \\ (distAgent(h_3, h_4), 0.67) \\ (distAgent(h_5, h_4), 0.54) \\ (timeout(h_5, h_3), 0.37) \\ (timeout(h_5, h_4), 0.38) \\ (timeout(h_3, h_4), 0.26) \\ (tileAvail(h_3, h_5), 0.08) \\ (tileAvail(h_4, h_5), 0.08) \\ (bt(W, Y) \leftarrow score(W, Y), 0.99) \\ (\sim bt(W, Y) \leftarrow score(Y, W), 0.99) \\ (bt(W, Y) \leftarrow distAgent(W, Y), 0.74) \\ (\sim bt(W, Y) \leftarrow distAgent(Y, W), 0.74) \\ (bt(W, Y) \leftarrow timeout(W, Y), 0.49) \\ (\sim bt(W, Y) \leftarrow timeout(Y, W), 0.49) \\ (bt(W, Y) \leftarrow tileAvail(W, Y), 0.24) \\ (\sim bt(W, Y) \leftarrow tileAvail(Y, W), 0.24) \end{array} \right\}$$

$$\Pi = \left\{ \begin{array}{l} (\sim bt(W, Y) \leftarrow sp(W, Y), 1) \\ (\sim bt(W, Y) \leftarrow sp(Y, W), 1) \end{array} \right\}$$

In the particular program presented above, the necessity degrees of the clauses belonging to (Π, Δ) were calculated as follows:

1. Normalize the alternatives' attribute values (Table 2) to interval $[0, 1]$ for all of the preference criteria (see Table 3).
2. Compare the alternatives among each other with respect to the normalized preference criteria. The alternative which appears as first argument of the comparison literal has a better attribute value (with respect to its associated preference criterion) than the one that appears as second argument. The necessity degree of the clause is calculated as the absolute value of the remainder of their normalized attribute values.
3. Divide the necessity degrees obtained in previous step by the number of preference criteria provided to the decision maker, *i.e.,* by 4 in this case.
4. Map the necessity degrees obtained in previous step to the subinterval assigned to the comparison literal in the clause (see Table 4).
5. For each clause (φ, α) such that φ is a rule of the kind $bt(W, Y) \leftarrow c_i(W, Y)$ or $\sim bt(W, Y) \leftarrow c_i(Y, W)$, set α to the upper bound value of the subinterval assigned to $c_i(\cdot, \cdot)$.

Criteria	Comparison literal	Subinterval
C_1	$score(\cdot, \cdot)$	$[0.75, 1)$
C_2	$timeout(\cdot, \cdot)$	$[0.250.5)$
C_3	$distAgent(\cdot, \cdot)$	$[0.500.75)$
C_4	$tileAvail(\cdot, \cdot)$	$[00.25)$

Table 1: Preferences per criterion

Alternatives	C_1	C_2	C_3	C_4
$h3$	3	8	2	2
$h4$	9	7	6	2
$h5$	6	15	5	3

Table 2: Alternatives values for each criterion

Alternatives	$C_{1\,[0,1]}$	$C_{2\,[0,1]}$	$C_{3\,[0,1]}$	$C_{4\,[0,1]}$
$h3$	0.33	0.53	0.33	0.67
$h4$	1	0.47	1	0.67
$h5$	0.67	1	0.83	1

Table 3: Alternatives values for each criterion normalized to $[0, 1]$

$(score(h_4, h_3), 0.67)$	$(timeout(h_3, h_4), 0.26)$
$(score(h_4, h_3), 0.17)$	$(distAgent(h_3, h_5), 0.5)$
$(score(h_4, h_3), 0.92)$	$(distAgent(h_3, h_5), 0.12)$
$(score(h_4, h_5), 0.33)$	$(distAgent(h_3, h_5), 0.62)$
$(score(h_4, h_5), 0.08)$	$(distAgent(h_3, h_4), 0.67)$
$(score(h_4, h_5), 0.83)$	$(distAgent(h_3, h_4), 0.17)$
$(score(h_5, h_3), 0.34)$	$(distAgent(h_3, h_4), 0.67)$
$(score(h_5, h_3), 0.08)$	$(distAgent(h_5, h_4), 0.17)$
$(score(h_5, h_3), 0.83)$	$(distAgent(h_5, h_4), 0.04)$
$(timeout(h_5, h_3), 0.47)$	$(distAgent(h_5, h_4), 0.54)$
$(timeout(h_5, h_3), 0.12)$	$(tileAvail(h_3, h_5), 0.33)$
$(timeout(h_5, h_3), 0.37)$	$(tileAvail(h_3, h_5), 0.08)$
$(timeout(h_5, h_4), 0.53)$	$(tileAvail(h_3, h_5), 0.08)$
$(timeout(h_5, h_4), 0.13)$	$(tileAvail(h_4, h_5), 0.33)$
$(timeout(h_5, h_4), 0.38)$	$(tileAvail(h_4, h_5), 0.08)$
$(timeout(h_3, h_4), 0.06)$	$(tileAvail(h_4, h_5), 0.08)$
$(timeout(h_3, h_4), 0.01)$	

Table 4: Alternatives comparison

The arguments of the form $\langle \mathcal{A}, h, \alpha \rangle$[3] presented in Figure 2(a), are built from the above P-DeLP conformant program. To calculate the accrued structures for these arguments, it will be used the ACC function defined below, with $K = 0.1$:[4]

$$ACC(\alpha_1, \ldots, \alpha_n) = [1 - \prod_{i=1}^{n}(1 - \alpha_i)] + K \max(\alpha_1, \ldots, \alpha_n) \prod_{i=1}^{n}(1 - \alpha_i)$$

As it can be observed in Figure 2(b), twelve a-structures can be built to support the reasons by which an alternative should be deemed better than another

[3]Given an argument $\langle \mathcal{A}, h, \alpha \rangle$, $\mathcal{A}$ is the set of uncertain clauses, h is the conclusion it supports and α its necessity degree.

[4]This function is a variant of the One-Complement accrual function used in [20] where K aims at weighting the importance given to the highest priority preference criterion, and in [11] has been proven to satisfy the desired properties to be used in multicriteria aggregation.

$$
\begin{array}{lll}
\mathcal{A}_1 = \langle & \{(bt(h_4,h_3) \leftarrow score(h_4,h_3), 0.99), (score(h_4,h_3), 0.92)\}, & bt(h_4,h_3), 0.92\rangle \\
\mathcal{A}_2 = \langle & \{(\sim bt(h_3,h_4) \leftarrow score(h_4,h_3), 0.99), (score(h_4,h_3), 0.92)\}, & \sim bt(h_3,h_4), 0.92\rangle \\
\mathcal{A}_3 = \langle & \{(bt(h_4,h_5) \leftarrow score(h_4,h_5), 0.99), (score(h_4,h_5), 0.83)\}, & bt(h_4,h_5), 0.83\rangle \\
\mathcal{A}_4 = \langle & \{(\sim bt(h_5,h_4) \leftarrow score(h_4,h_5), 0.99), (score(h_4,h_5), 0.83)\}, & \sim bt(h_5,h_4), 0.83\rangle \\
\mathcal{A}_5 = \langle & \{(bt(h_5,h_3) \leftarrow score(h_5,h_3), 0.99), (score(h_5,h_3), 0.83)\}, & bt(h_5,h_3), 0.83\rangle \\
\mathcal{A}_6 = \langle & \{(\sim bt(h_3,h_5) \leftarrow score(h_5,h_3), 0.99), (score(h_5,h_3), 0.83)\}, & \sim bt(h_3,h_5), 0.83\rangle \\
\mathcal{A}_7 = \langle & \{(bt(h_3,h_5) \leftarrow distAgent(h_3,h_5), 0.74), (distAgent(h_3,h_5)), 0.62)\}, & bt(h_3,h_5), 0.62\rangle \\
\mathcal{A}_8 = \langle & \{(\sim bt(h_5,h_3) \leftarrow distAgent(h_3,h_5), 0.74), (distAgent(h_3,h_5)), 0.62)\}, & \sim bt(h_5,h_3), 0.62\rangle \\
\mathcal{A}_9 = \langle & \{(bt(h_3,h_4) \leftarrow distAgent(h_3,h_4), 0.74), (distAgent(h_3,h_4), 0.67)\}, & bt(h_3,h_4), 0.67\rangle \\
\mathcal{A}_{10} = \langle & \{(\sim bt(h_4,h_3) \leftarrow distAgent(h_3,h_4), 0.74), (distAgent(h_3,h_4), 0.67)\}, & \sim bt(h_4,h_3), 0.67\rangle \\
\mathcal{A}_{11} = \langle & \{(bt(h_5,h_4) \leftarrow distAgent(h_5,h_4), 0.74), (distAgent(h_5,h_4), 0.54)\}, & bt(h_5,h_4), 0.54\rangle \\
\mathcal{A}_{12} = \langle & \{(\sim bt(h_4,h_5) \leftarrow distAgent(h_5,h_4), 0.74), (distAgent(h_5,h_4), 0.54)\}, & \sim bt(h_4,h_5), 0.54\rangle \\
\mathcal{A}_{13} = \langle & \{(bt(h_5,h_3) \leftarrow timeout(h_5,h_3), 0.49), (timeout(h_5,h_3), 0.37)\}, & bt(h_5,h_3), 0.37\rangle \\
\mathcal{A}_{14} = \langle & \{(\sim bt(h_3,h_5) \leftarrow timeout(h_5,h_3), 0.49), (timeout(h_5,h_3), 0.37)\}, & \sim bt(h_3,h_5), 0.37\rangle \\
\mathcal{A}_{15} = \langle & \{(bt(h_5,h_4) \leftarrow timeout(h_5,h_4), 0.49), (timeout(h_5,h_4), 0.38)\}, & bt(h_5,h_4), 0.38\rangle \\
\mathcal{A}_{16} = \langle & \{(\sim bt(h_4,h_5) \leftarrow timeout(h_5,h_4), 0.49), (timeout(h_5,h_4), 0.38)\}, & \sim bt(h_4,h_5), 0.38\rangle \\
\mathcal{A}_{17} = \langle & \{(bt(h_3,h_4) \leftarrow timeout(h_3,h_4), 0.49), (timeout(h_3,h_4), 0.26)\}, & bt(h_3,h_4), 0.26\rangle \\
\mathcal{A}_{18} = \langle & \{(\sim bt(h_4,h_3) \leftarrow timeout(h_3,h_4), 0.49), (timeout(h_3,h_4), 0.26)\}, & \sim bt(h_4,h_3), 0.26\rangle \\
\mathcal{A}_{19} = \langle & \{(bt(h_3,h_5) \leftarrow tileAvail(h_3,h_5), 0.24), (tileAvail(h_3,h_5), 0.08)\}, & bt(h_3,h_5), 0.08\rangle \\
\mathcal{A}_{20} = \langle & \{(\sim bt(h_5,h_3) \leftarrow tileAvail(h_3,h_5), 0.24), (tileAvail(h_3,h_5), 0.08)\}, & \sim bt(h_5,h_3), 0.08\rangle \\
\mathcal{A}_{21} = \langle & \{(bt(h_4,h_5) \leftarrow tileAvail(h_4,h_5), 0.24), (tileAvail(h_4,h_5), 0.08)\}, & bt(h_4,h_5), 0.08\rangle \\
\mathcal{A}_{22} = \langle & \{(\sim bt(h_5,h_4) \leftarrow tileAvail(h_4,h_5), 0.24), (tileAvail(h_4,h_5), 0.08)\}, & \sim bt(h_5,h_4), 0.08\rangle
\end{array}
$$

(a)

$$
\begin{array}{llll}
[\Phi_1, bt(h_3,h_5), 0.67], & [\mathbf{\Phi'_1}, \sim \mathbf{bt(h_3,h_5)}, \mathbf{0.90}], & \Phi_1 = \mathcal{A}_7 \cup \mathcal{A}_{19}, & \Phi'_1 = \mathcal{A}_6 \cup \mathcal{A}_{14}; \\
[\Phi_2, \sim bt(h_5,h_3), 0.67], & [\mathbf{\Phi'_2}, \mathbf{bt(h_5,h_3)}, \mathbf{0.90}], & \Phi_2 = \mathcal{A}_8 \cup \mathcal{A}_{20}, & \Phi'_2 = \mathcal{A}_5 \cup \mathcal{A}_{13}; \\
[\Phi_3, bt(h_3,h_4), 0.78], & [\mathbf{\Phi'_3}, \sim \mathbf{bt(h_3,h_4)}, \mathbf{0.93}], & \Phi_3 = \mathcal{A}_9 \cup \mathcal{A}_{17}, & \Phi'_3 = \mathcal{A}_2; \\
[\Phi_4, \sim bt(h_4,h_3), 0.78], & [\mathbf{\Phi'_4}, \mathbf{bt(h_4,h_3)}, \mathbf{0.93}], & \Phi_4 = \mathcal{A}_{10} \cup \mathcal{A}_{18}, & \Phi'_4 = \mathcal{A}_1; \\
[\Phi_5, bt(h_5,h_4), 0.73], & [\mathbf{\Phi'_5}, \sim \mathbf{bt(h_5,h_4)}, \mathbf{0.85}], & \Phi_5 = \mathcal{A}_{11} \cup \mathcal{A}_{15}, & \Phi'_5 = \mathcal{A}_4 \cup \mathcal{A}_{22}; \\
[\Phi_6, \sim bt(h_4,h_5), 0.73], & [\mathbf{\Phi'_6}, \mathbf{bt(h_4,h_5)}, \mathbf{0.85}], & \Phi_6 = \mathcal{A}_{12} \cup \mathcal{A}_{16}, & \Phi'_6 = \mathcal{A}_3 \cup \mathcal{A}_{21};
\end{array}
$$

(b)

Figure 2: Arguments and a-structures built from the P-DeLP conformant program

one. Those a-structures warranted from the dialectical process (shown in bold), will be used by Algorithm 5 to compute the set of acceptable alternatives. In this particular case, only decision rule DR1 can be applied. For alternative h_4, precondition of DR1 can be warranted and like there is no warranted a-structure supporting a conclusion of the kind $bt(Z, h_4)$ to warrant DR1's restriction, h_4 becomes the acceptable alternative. Finally, hole h_4 becomes IH.

5.2. Means-ends Reasoning and Intention Reconsideration

Once a hole has been selected to fill in, plans to achieve this intention are selected. The criteria set provided for plan selection could be $C = \{C_1, C_2, C_3\}$, where $C_1 = length$, $C_2 = cost$, $C_3 = timeoutTile$. Criterion C_1 is the number of action within the plan. C_2 represents the plan cost which is calculated as the sum of its actions costs, which depend on the agent's orientation. Finally, C_3 is the time-out time of the tile selected in the plan to fill in the hole.

On the other hand, the fact that holes appear and disappear causes the agent to change its intentions. For example, when the set of holes dot not change while the agent is executing a plan, then there is no need to deliberate; but if the set of holes do change, this might mean that IH has disappeared or that a closer hole has appeared; thus, intentions reconsideration is necessary. To achieve this behaviour, it is important to consider appropriate criteria to determine whether these changes have occurred or not. Means-ends reasoning and intention reconsideration also use the argumentation-based decision framework (as in the filtering stage), in order to choose a plan to execute or to reconsider intentions, while the plan is under execution. Due to space constraints, how this framework is applied in these stages, will not be developed in this paper.

6. CONCLUSIONS

In this work, we presented an abstract framework that integrates argumentation-based decision making from a multi-criteria approach, within the inner decision

processes of a BDI agent. Likewise, it was specified how to perform a concrete implementation of the inner decision making processes within a BDI agent.

In order to get a better understanding and provide feedback to the abstraction process carried out to propose this present framework, as future work, following the idea proposed in [21], new instantiations of the framework will be done with other methods belonging to the research field of Agreement Technologies.

7. ACKNOWLEDGMENTS

This work was supported by Universidad Nacional de San Luis (PROICO 30312).

8. REFERENCES

[1] M. Bratman, *Intentions, Plans and Practical Reason*. Cambridge, MA: Harvard University Press, 1987.

[2] M. Bratman, D. Israel, and M. Pollack, "Plans and resource bounded reasoning," *Computational Intelligence*, vol. 4, no. 4, pp. 349–355, 1988.

[3] D. C. Dennett, "Intentional systems," *Journal of Philosophy*, vol. 68, pp. 87–106, 1971.

[4] M. Ljungberg and A. Lucas, "The oasis air traffic management system," Tech. Rep. 28, Civil Aviation of Australia, August 1992.

[5] A. S. Rao, A. Lucas, D. Morley, M. Selvestrel, and G. Murray, "Agent-oriented architecture for air-combat simulation," Tech. Rep. 43, Australian Artificial Intelligence Institute, 1993.

[6] R. Evertsz, M. Fletcher, R. Jones, J. Jarvis, J. Brusey, and S. Dance, *Programming Multi-Agent Systems*, ch. Implementing Industrial Multi-agent Systems Using JACK. Springer, 2004.

[7] S. S. Benfield, J. Hendrickson, and D. Galanti, "Making a strong business case for multiagent technology," in *5th AAMAS*, 2006.

[8] A. Casali, L. Godo, and C. Sierra, "A graded BDI agent model to represent and reason about preferences," *Artifical Intelligence*, vol. 175, no. 7-8, pp. 1468–1478, 2011.

[9] L. Amgoud, "A formal framework for handling conflicting desires," in *ECSQARU* (T. D. Nielsen and N. L. Zhang, eds.), vol. 2711 of *Lecture Notes in Computer Science*, pp. 552–563, Springer, 2003.

[10] L. Amgoud and H. Prade, "Formalizing practical reasoning under uncertainty: An argumentation-based approach," in *IAT*, pp. 189–195, IEEE Computer Society, 2007.

[11] E. Ferretti, M. Errecalde, A. García, and G. Simari, "A possibilistic defeasible logic programming approach to argumentation-based decision making," *Journal of Experimental & Theoretical Artificial Intelligence*, 2014. In press. Draft version at `https://sites.google.com/site/edgardoferretti/TETA-2012-0093.R1.pdf?attredirects=0&d=1`.

[12] N. D. Rotstein, A. J. García, and G. R. Simari, "Reasoning from desires to intentions: A dialectical framework," in *AAAI*, pp. 136–141, AAAI Press, 2007.

[13] F. Schlesinger, E. Ferretti, M. Errecalde, and G. Aguirre, "An argumentation-based BDI personal assistant," in *IEA/AIE*, vol. 6069 of *LNAI*, Springer, 2010.

[14] A. García and G. Simari, "Defeasible logic programming: an argumentative approach," *Theory and Practice of Logic Programming*, vol. 4, no. 2, pp. 95–138, 2004.

[15] M. Wooldridge, *Reasoning about Rational Agents*. The MIT Press, 2000.

[16] D. N. Kinny, "Commitment and effectiveness of situated agents," in *In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91*, pp. 82–88, 1991.

[17] M. Schut and M. Wooldridge, "Intention reconsideration in complex environments," in *4th International Conference on Autonomous Agents*, 2000.

[18] S. Russell, E. Wefald, M. Karnaugh, R. Karp, D. Mcallester, D. Subramanian, and M. Wellman, "Principles of metareasoning," *Artificial Intelligence*, 1991.

[19] M. E. Pollack and M. Ringuette, "Introducing the tileworld: Experimentally evaluating agent architectures," in *8th AAAI*, pp. 183–189, 1990.

[20] M. Gómez, C. Chesñevar, and G. Simari, "Modelling argument accrual in possibilistic defeasible logic programming," in *ECSQARU*, LNCS, pp. 131–143, Springer, 2009.

[21] C. Sosa-Toranzo, F. Schlesinger, E. Ferretti, and M. Errecalde, "Integrating a voting protocol within an argumentation-based BDI system," in *XVI CACIC*, 2010.

Received: December 2013. Accepted: February 2014.