Programming BOID-Plan Agents deliberating about conflicts among defeasible mental attitudes and plans

Mehdi Dastani
Utrecht University
mehdi@cs.uu.nl

Leendert van der Torre
CWI
torre@cwi.nl

Abstract

In this paper we present an abstract agent programming language and its operational semantics which can be used to implement cognitive agents. This language consists of programming constructs to implement both the agent's mental attitudes – interpreted as data structures – as well as the agent's deliberation process. The agent can observe the environment, generate goal sets from desires, obligations, and intentions, selects goals, generate plans, and execute them. These actions can be combined in the deliberation language in a variety of ways to program the agent's deliberation process. At the level of abstraction of our deliberation language, goal generation and planning are both characterized as conflict resolution procedures. For goal generation, obligation, desire and intention rules can conflict when the corresponding goals are incompatible. For planning, partial plans can be incompatible. In our approach, the incompatibility of plans can be derived from more detailed data structures such as resources of the agents, but the conflict procedure can also be programmed directly by the agent programmer.

1. Introduction

Agent programming languages need explicit mechanisms to balance mental attitudes like beliefs, desires, goals, intentions and plans as studied in agent specification languages [6, 10, 11] and characterized by for example properties of realism and commitment strategies. In this paper we generalize our earlier work on 3APL [7], which is based on refinement planning but neither incorporates goal generation, nor a goal selection or plan selection mechanism. The framework of classical planning is not sufficient to study the balance in the context of agent programming, because the planning algorithm is only a part of the deliberation process of such agents, and planning must be interleaved with for example observations and intention reconsideration [2, 3]. The framework of decision-theoretic planning seems better suited to model the balance, but it is not easily related to agent implementation languages, because the development of tools to study the balance between mental attitudes in this setting is still unexplored [8].

We envision a programming language with primitives such as observations of the environment, generation of goal sets from desires, obligations, and intentions, selection of goals, generation of plans, and execution of plans. We call such a programming language a *deliberation language* [7, 8] to emphasize that it can be used to program an agent deliberation process. Some of these primitives are motivated by our earlier work on BOID agent architecture [4], though similar mechanisms can be found in other languages and architectures. The BOID architecture combines goal generation and planning in a fixed (non-programmable) deliberation cycle. The problem of defining an agent deliberation language with operational semantics breaks down in the following research questions.

1. How do we model the mental attitudes? In our setting, inspired by the BOID architecture, the data structures are defeasible mental attitudes represented by rules.

2. How do we characterize the primitive instructions at the level of abstraction of a deliberation language? We identify goal generation and planning as two primitive instructions, which are both characterized as conflict resolution procedures. In this paper we do not detail the planning algorithm of the agent, such that also other planning algorithms can be incorporated.

3. How do we combine primitive instructions in a deliberation language?

The layout of this paper is as follows. In Section 2 we discuss the role of conflict resolution, and introduce a running example. In Section 3 we discuss the syntax of the deliberation language, and in Section 4 we discuss its operational semantics.

2. Deliberation

We assume a separation between mental attitudes (data structures) and deliberation instructions (programming instructions) that manipulate the mental attitudes. In our setting, inspired by the BOID architecture, the data structures are defeasible mental attitudes represented by rules. In other programming languages mental attitudes are not defeasible and not modelled as rules but for example as a set of formulae closed under logical consequence.

The primitive instructions goal generation and planning are both characterized as conflict resolution procedures.

- For goal generation, obligation, desire and intention rules can conflict when the corresponding goals are incompatible.
- For planning, partial plans can be incompatible. In our agent architecture, the incompatibility of plans can be derived from more detailed data structures such as resources of the agents, but the conflict procedure can also be programmed directly by the agent programmer.

In this paper we show how these primitives can be combined using standard programming constructs such as tests and choice and iteration instructions. In particular, this enables the programmer to implement a balance between goal generation and planning. For example, a deliberation program may consists of two conditional iterations (while-loops) such that the condition of the first holds as long as there is no emergency situation while the condition of the second holds as long as there is an emergency situation. Moreover, the body of the first iteration generates goals and then plans them, while the body of the second can generate emergency plans and execute them. This example shows that our language is expressive enough to implement important aspects of subsumption architectures [5], in which emergency behavior can be realized at the reactive layer while complex behavior can be realized at higher deliberative layers.

2.1. Conflicts: reasoning or programming?

The agent deliberation cycle can generate many goals, and for each goal it can generate many plans. It is crucial to prune the number of generated goals and plans as soon as possible. One way to do so is to check whether goals are feasible, and whether plans are coherent, non-conflicting, etc. We leave the determination of conflicts between plans to the programmer. Programming conflicts explicitly also leads to a more general and more flexible notion of plan rules. Below is a non-exhaustive list of conflict patterns the programmer may use.

Consistency. An agent may have a goal for p and a goal for $\neg p$, stating that he should see to both p and $\neg p$ (though clearly at different times!). However, the programmer can also decide to eliminate such goals.

Complexity. Various measures can be defined, such as for example a maximum of five actions in a plan.

Resource constraints. The usual approach is to define the resources of a plan, and check whether these resources do not exceed the available set of resources. There are several sophisticated algorithms available, which can be incorporated. However, there are also drawbacks to this approach. It is typically complex and expensive to specify all resources needed in applications. Moreover, it is hard to debug programs that contain errors in the specified sets of resources.

Temporal constraints. Although time may be seen as a resource itself, it is a resource with particular properties, as they have been studied in scheduling. E.g., how to include forecasts of the agent's deliberation time?

2.2. Example

The following example will be used in the remainder of the paper. Agent ag believes that it is the time to go on holiday or to the AAMAS'04 conference, desires to go to New York or San Francisco if he believes it is time to go on holiday, desires to go to New York for the conference and has the obligation to pay the registration fee, if he believes it is conference time. Moreover, agent ag has specific (partial) plans to achieve its desires. In particular, its (partial) plans to realize its desires to go to New York or San Francisco is to desire first to have a ticket to New York or San Francisco, respectively, followed by a desire to have a hotel room in either of these cities. Finally, the plan of agent ag to satisfy its obligation to pay the conference fee is to transfer the registration fee, its plan to achieve its desires to have a ticket is to buy tickets from either KLM or by British Air, and its plan to achieve its desire to have a hotel room in those cities is to contact hotel1 and hotel2 in these cities.

In this example, we assume that if ag believes that it is a holiday time, then it does not believe that it is the conference time and vise versa. Similarly, if the agent wants to go to New York, then it does not want to go to San Francisco. As it will be shown, these assumptions can be formally modelled as integrity constraints on agent's beliefs and goals. Given these integrity constraints, many plans can be generated. However, since the agent has a limited amount of money (resources), many of the possible plans become incoherent. We define the coherence of a plan in terms of resource constraint. In particular, we assume that a plan is coherent if its costs is lower or equal to $\$2000$.

3. BOID: Syntax

We define languages to specify the agent's mental attitudes. We allow modal formulae with iterated modal operators to describe the mental state, though in practical applications we intend to use a fragment of this language, for example based on literals and a bounded number of nested modalities.

Definition 1 *(domain language) Let $P = \{p, q, \ldots\}$ be a set of propositional atoms, and L_P be the propositional language built up from these atoms in the usual way. The language L is defined as follow:*

- $L_P \subseteq L$
- *if $\phi \in L$, then $\mathbf{B}(\phi), \mathbf{O}(\phi), \mathbf{I}(\phi), \mathbf{D}(\phi) \in L$*
- *if $\phi, \psi \in L$, then $\neg\phi, \phi \wedge \psi \in L$.*

The following definition introduces the language for partial plans. Obligation, intention, and desire formulae are considered as plans as well. A plan that contains such a formula is called a partial plan, because the agent still has to refine this plan before it can execute it. In a refinement step, the formula is replaced by another partial plan.

Definition 2 *(plan language) Let $Act = \{\alpha, \beta, \ldots\}$ be a set of basic actions, $\phi \in L$, and $\psi \in \{\mathbf{O}(\phi), \mathbf{I}(\phi), \mathbf{D}(\phi) \mid \phi \in L\}$ be a goal (i.e. an obligation, intention, or desire). The plan language L_{plan} is defined as follows:*

- $Act \subseteq L_{plan}$ *(basic action)*
- $\phi? \in L_{plan}$ *(test action)*
- $\psi \in L_{plan}$ *(partial plan)*
- $\epsilon \in L_{plan}$ *(empty action)*
  We assume $\forall\lambda \in L_{plan} : \epsilon;\lambda = \lambda;\epsilon = \lambda$.
- *if $\lambda, \lambda' \in L_{plan}$, then $\lambda;\lambda' \in L_{plan}$ (sequence)*
- *if $\lambda, \lambda' \in L_{plan}$, then*
  - $\texttt{while}\ \phi\ \texttt{do}\ \lambda\ \texttt{od} \in L_{plan}$ *(iteration)*
  - $\texttt{if}\ \phi\ \texttt{then}\ \lambda\ \texttt{else}\ \lambda'\ \texttt{fi} \in L_{plan}$ *(conditional choice)*

Given the languages to specify agent's mental states, we specify BOID agents. A BOID agent consists of belief, obligation, desire, intention and planning rules, where we write $a \hookrightarrow b$ for the rule 'if a then b'.

Definition 3 *(agent specification) A BOID agent is specified as a tuple $\Delta = \langle \mathcal{B}, \mathcal{O}, \mathcal{I}, \mathcal{D}, \mathcal{P}, \rho \rangle$ where:*

- *$\mathcal{B} \subseteq \{\phi \hookrightarrow \mathbf{B}(\psi) \mid \phi, \psi \in L\}$ represents agent's belief rules.*
- *$\mathcal{O} \subseteq \{\phi \hookrightarrow \mathbf{O}(\psi) \mid \phi, \psi \in L\}$ represents agent's obligation rules.*
- *$\mathcal{I} \subseteq \{\phi \hookrightarrow \mathbf{I}(\psi) \mid \phi, \psi \in L\}$ represents agent's intention rules.*
- *$\mathcal{D} \subseteq \{\phi \hookrightarrow \mathbf{D}(\psi) \mid \phi, \psi \in L\}$ represents agent's desire rules.*
- *$\mathcal{P} \subseteq \{\phi \hookrightarrow \lambda \mid \phi \in L \backslash L_P, \lambda \in L_{plan}\}$ represents agent's planning rules.*
- *ρ is a totally connected reflexive and transitive relation on the set of rules, called the priority relation.*

Consider again the example of agent ag described in section 2.2. Let the strings starting with a lower-case letter be propositions from L and strings starting with a capital letter be an action. The agent ag can be specified as $\Delta_{ag} = \langle \mathcal{B}, \mathcal{O}, \mathcal{I}, \mathcal{D}, \mathcal{P}, \rho \rangle$ where the priority order on the rules is such that belief rules have the highest priority (representing a kind of realism, see [4]) and desire rules have a higher priority than obligation rules (representing a kind of selfishness [4]):

$\mathcal{B} = \{\top \hookrightarrow \mathbf{B}(timeforholiday),$
$\quad \top \hookrightarrow \mathbf{B}(timeforconference)\},$
$\mathcal{O} = \{\mathbf{B}(timeforconference) \hookrightarrow \mathbf{O}(payregistration)\}$
$\mathcal{I} = \emptyset$
$\mathcal{D} = \{\mathbf{B}(timeforholiday) \hookrightarrow \mathbf{D}(gotoNewYork),$
$\quad \mathbf{B}(timeforholiday) \hookrightarrow \mathbf{D}(gotoSanFransisco),$
$\quad \mathbf{B}(timeforconference) \hookrightarrow \mathbf{D}(gotoNewYork)\},$
$\mathcal{P} = \{\mathbf{D}(gotoNewYork) \hookrightarrow \mathbf{D}(haveNYticket); \mathbf{D}(haveroomNY),$
$\quad \mathbf{D}(gotoSanFransisco) \hookrightarrow \mathbf{D}(haveSFticket); \mathbf{D}(haveroomSF),$
$\quad \mathbf{O}(payregistration) \hookrightarrow TransferMoney,$
$\quad \mathbf{D}(haveNYticket) \hookrightarrow BuyKLMticketNY,$
$\quad \mathbf{D}(haveNYticket) \hookrightarrow BuyBritishAirticketNY,$
$\quad \mathbf{D}(haveSFticket) \hookrightarrow BuyKLMticketSF,$
$\quad \mathbf{D}(haveSFticket) \hookrightarrow BuyBritishAirticketSF,$
$\quad \mathbf{D}(haveroomNY) \hookrightarrow ContactHotel1NY,$
$\quad \mathbf{D}(haveroomNY) \hookrightarrow ContactHotel2NY,$
$\quad \mathbf{D}(haveroomSF) \hookrightarrow ContactHotel1SF,$
$\quad \mathbf{D}(haveroomSF) \hookrightarrow ContactHotel2SF\},$
$\forall r_b \in \mathcal{B}, r_d \in \mathcal{D}, r_o \in \mathcal{O} : \rho(r_b) > \rho(r_d) > \rho(r_o).$

4. BOID: Semantics

A BOID agent, specified in terms of its mental attitudes, has a mental state at each moment of time depending on the input it receives and its deliberation process. The state of a BOID agent consists of a set of facts (F) observed from the environment, which constitutes agent's observations, a belief base (σ) that constitutes agent's beliefs, a goal base (γ) that constitutes agent's objectives, an intention base (δ) that constitutes agent's (new) intentions, and a plan base (π) that constitutes agent's ways to achieve objectives. The set A is the union of the goal base and intention base, thus a set of sets of **B**, **O**, **I**, and **D** formulas.

Definition 4 *(Agent state) The state of a BOID agent is a tuple $\mathcal{S} = \langle F, A, \sigma, \gamma, \delta, \pi \rangle$, where:*

- *$F \subseteq L_P$ is a set of facts;*

- $A \subseteq Pow(\{\mathbf{B}(\phi), \mathbf{O}(\phi), \mathbf{I}(\phi), \mathbf{D}(\phi) \mid \phi \in L\})$.
- $\sigma = \{E \cap \{\mathbf{B}(\phi) \mid \phi \in L\} \mid E \in A\}$ *(beliefbase)*
- $\gamma = \{E \cap \{\mathbf{O}(\phi), \mathbf{I}(\phi), \mathbf{D}(\phi) \mid \phi \in L\} \mid E \in A\}$ *(goalbase)*
- $\delta \subseteq \gamma$ *(intentionbase)*
- $\pi \subseteq Pow(L_{plan})$ *(planbase)*

Definition 6 uses normal default rules to specify agent's mental attitudes, and sets of extensions represent agent's mental states such as beliefs and goals. Definition 5 explains that a rule $a \hookrightarrow b$ can be applied in context E when E implies a, but it does not imply b or $\neg b$. If there are more than one rule which can be applied, then we apply a rule with the highest priority in the priority ordering ρ; these rules are denoted in the following definitions by $\max(E, \mathcal{S}, R)$.

Definition 5 *Let $\mathcal{S}$ be a BOID state, ρ be a priority relation, an extension E be a consistent set of L formulas, and $\models$ the satisfiability relation interpreting the modal operators of L as normal modal operators (see e.g. [1]).*

- *a rule $(\phi \hookrightarrow \psi)$ is strictly applicable to an extension E in $\mathcal{S}$, iff $E \models \phi$, $E \not\models \psi$ and $E \not\models \neg\psi$*
- *$\max(E, \mathcal{S}, R)$ is the set of rules $(\phi \hookrightarrow \psi)$ from R that are strictly applicable to E in $\mathcal{S}$ such that there does not exists a $(\phi' \hookrightarrow \psi') \in R$ strictly applicable to E in $\mathcal{S}$ with $\rho(\phi' \hookrightarrow \psi') > \rho(\phi \hookrightarrow \psi)$*

Definition 6 defines the belief and goal states by iteratively applying the rules with the highest priority. In particular, we define agent's belief state in terms of its observation state and its belief rules, and we define the goal state in terms of its belief, obligation, desire, and intention rules. The observation state, the intention state, and the plan state should then be determined by means of agent deliberation as we explain after the following definition.

Definition 6 *Let $\Delta = \langle \mathcal{B}, \mathcal{O}, \mathcal{I}, \mathcal{D}, \mathcal{P}, \rho \rangle$ be the specification of a BOID agent, and $\mathcal{S} = \langle F, A, \sigma, \gamma, \delta, \pi \rangle$ a state of the agent. Given Δ and $\mathcal{S}$, the belief extension, $BelExt(\Delta, \mathcal{S})$, and the goal extension, $GoalExt(\Delta, \mathcal{S})$, of agent Δ in state $\mathcal{S}$ are specified as follows:*

- *$BelExt(\Delta, \mathcal{S}) = \langle F, A_i, \sigma_i, \gamma, \delta, \pi \rangle$ iff $A_i = A_{i-1}$ for the following procedure:*

  $i := 0$;
  $A_i := A$;
  repeat
    $A_{i+1} := \emptyset$;
    for all $E \in A_i$ ***do***
      if *exists* $(\phi \hookrightarrow \psi) \in \mathcal{B}$ *strictly applicable to* E *in* $\langle F, A_i, \sigma_i, \gamma, \delta, \pi \rangle$ ***then***
        for all $(\phi \hookrightarrow \psi) \in \max(E, \langle F, A_i, \sigma_i, \gamma, \delta, \pi \rangle, \mathcal{B})$ ***do***
          $A_{i+1} := A_{i+1} \cup \{ E \cup \psi \}$;
        end for
      else
        $A_{i+1} := A_{i+1} \cup \{E\}$;
      end if
    end for
    i:=i+1;
  until $A_i = A_{i-1}$;

- *$GoalExt(\Delta, \mathcal{S}) = \langle F, A_i, \sigma_i, \gamma_i, \delta, \pi \rangle$ iff $A_i = A_{i-1}$ for the following procedure:*

  $i := 0$;
  $A_i := A$;
  repeat
    $A_{i+1} := \emptyset$;
    for all $E \in A_i$ ***do***
      if *exists* $(\phi \hookrightarrow \psi) \in \mathcal{B} \cup \mathcal{O} \cup \mathcal{I} \cup \mathcal{D}$ *strictly applicable to* E *in* $\langle F, A_i, \sigma_i, \gamma_i, \delta, \pi \rangle$ ***then***
        for all $(\phi \hookrightarrow \psi) \in \max(E, \langle F, A_i, \sigma_i, \gamma_i, \delta, \pi \rangle, \mathcal{B} \cup \mathcal{O} \cup \mathcal{I} \cup \mathcal{D})$ ***do***
          $A_{i+1} := A_{i+1} \cup \{ E \cup \psi \}$;
        end for
      else
        $A_{i+1} := A_{i+1} \cup \{E\}$;
      end if
    end for
    i:=i+1;
  until $A_i = A_{i-1}$;

Note in the above algorithms that for each applicable rule in max one extension is added. Consider again the example of agent ag from section 2.2 and its specification Δ_{ag} as given in section 3. Then, the goals of the agent ag at the initial state $\mathcal{S} = \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$ is indicated by $GoalExt(\Delta_{ag}, \mathcal{S}) = \langle \emptyset, A, \sigma, \gamma, \emptyset, \emptyset \rangle$, where:

$A = \{\{B(timeforholiday), D(gotoNewYork)\},$
$\{B(timeforholiday), D(gotoSanFransisco)\},$
$\{B(timeforconference), D(gotoNewYork),$
$O(payregistration)\}\},$
$\sigma = \{\{B(timeforholiday)\}, \{B(timeforconference)\}\},$
$\gamma = \{\{D(gotoNewYork)\}, \{D(gotoSanFransisco)\},$
$\{D(gotoNewYork), O(payregistration)\}\}$

Definition 5 and 6 assume that rules are applied for each extension separately. Alternatively, a more global rule application mechanism can be defined. For example, we can use a rule application mechanism in which the clause $E \models \phi$ in the first item of definition 5 is replaced by $\mathcal{S} \models \phi$ for a suitable definition of $\models$. Definition 7 distinguishes between a credulous and sceptical variant.

Definition 7 *Let $\mathcal{S} = \langle F, A, \sigma, \gamma, \delta, \pi \rangle$ be a BOID state. The credulous entailment relation between state and formulas is defined as follows:*

- $\mathcal{S} \models \mathbf{B}(\phi) \Leftrightarrow \exists E \in \sigma : E \models \mathbf{B}(\phi)$

- $\mathcal{S} \models \mathbf{X}(\phi) \Leftrightarrow \exists E \in \gamma : E \models \mathbf{X}(\phi)$ *for* $\mathbf{X} \in \{\mathbf{O}, \mathbf{I}, \mathbf{D}\}$

A cautious entailment relation is defined as follows:

- $\mathcal{S} \models \mathbf{B}(\phi) \Leftrightarrow \forall E \in \sigma : E \models \mathbf{B}(\phi)$
- $\mathcal{S} \models \mathbf{X}(\phi) \Leftrightarrow \forall E \in \gamma : E \models \mathbf{X}(\phi)$ *for* $\mathbf{X} \in \{\mathbf{O}, \mathbf{I}, \mathbf{D}\}$

An agent commits itself to a subset of the generated goal extensions. The set of committed goals is called intention. The selection of the goals is done by a selection function at the deliberation level as we will see in the next section. Once the intentions are generated, they need to be planned. Given the intentions, the set of plans that can achieve them can be constructed by generating plan extensions based on the set of planning rules. A plan extension is generated in a similar way as belief and goal extensions. The only difference is that the plan extension can be extended with the consequent of a planning rule, which is a plan, if this plan does not make the plan extension incoherent. Note that the belief and goal extensions could be extended if the extension did not become inconsistent. For plan extension we do not use the notion of consistency, but rather the notion of coherence which can be defined, for example, in terms of resource conflicts or possibility of its execution. The idea is that the plan extension can be considered as one (parallel or sequential) plan in the same way as a goal extension was considered as one (conjunctive) goal. Extending a plan extension with a plan means then extending a plan. Definition 8 extends a plan extension with a plan.

Definition 8 *Let* $occurs(\phi, \lambda)$ *be a sentence which is true if and only if the plan* λ *is a partial plan and the goal* ϕ *occurs in* λ*. Let also* $sub(\lambda', \phi, \lambda)$ *be the operation which substitutes the plan* λ *for the goal* ϕ *that occurs in the plan* λ'*. Finally, let* E *be a plan extension, i.e., a set of plans. The plan extension* E *can then be extended with a plan* λ*, which achieves* ϕ*, denoted by* $add(E, \lambda, \phi)$*, as follows:*

$$add(E, \lambda, \phi) =$$
$$\begin{array}{ll} - \; E \backslash \{\lambda'\} \cup \{sub(\lambda', \phi, \lambda)\} & \text{if } \exists \lambda' \in E : \; occurs(\phi, \lambda') \\ - \; E \cup \lambda & otherwise \end{array}$$

The extended plan should be coherent which can be defined in various ways in terms of resource conflicts, etc. In the following, we assume a decision method that decides whether a plan is coherent.

Definition 9 *Let* $\Delta = \langle \mathcal{B}, \mathcal{O}, \mathcal{I}, \mathcal{D}, \mathcal{P}, \rho \rangle$ *be a BOID agent and* $\mathcal{S} = \langle F, A, \sigma, \gamma, \delta, \pi \rangle$ *be a BOID state, a plan extension* $E \in \pi$ *be a set of plan expressions, and* $coherent(\lambda)$ *will be a predicate that is true when the plan* λ *is coherent.*

- *a planning rule* $(\phi \hookrightarrow \lambda)$ *is strictly applicable to a plan extension* E *in* $\mathcal{S}$*, iff* $\delta \models \phi$ *or* $\exists \lambda' \in E :$ $occurs(\phi, \lambda')$ *and* $coherent(add(E, \lambda, \phi))$*.*
- $\max(E, \mathcal{S}, R)$ *is the set of planning rules* $(\phi \hookrightarrow \lambda)$ *from* R *that are strictly applicable to plan extension* E *in* $\mathcal{S}$ *such that there does not exists a* $(\phi' \hookrightarrow \lambda') \in R$ *strictly applicable to* E *in* $\mathcal{S}$ *with* $\rho(\phi' \hookrightarrow \lambda') > \rho(\phi \hookrightarrow \lambda)$

We now define agent's plan state in terms of its goal state.

Definition 10 *Let* $\Delta = \langle \mathcal{B}, \mathcal{O}, \mathcal{I}, \mathcal{D}, \mathcal{P}, \rho \rangle$ *be the specification of a BOID agent, and* $\mathcal{S} = \langle F, A, \sigma, \gamma, \delta, \pi \rangle$ *a state of the agent. Given* Δ *and* $\mathcal{S}$*, the plan extension,* $planExt(\Delta, \mathcal{S})$*, of agent* Δ *in state* $\mathcal{S}$ *is specified as follows:*

$PlanExt(\Delta, \mathcal{S}) = \langle F, A, \sigma, \gamma, \delta, \pi_i \rangle$ *iff* $\pi_i = \pi_{i-1}$ *for the following procedure:*

$i := 0$*;*
$\pi_i := \pi$*;*
repeat
  $\pi_{i+1} := \emptyset$*;*
  for all $E \in \pi_i$ ***do***
    if exists $(\phi \hookrightarrow \lambda) \in \mathcal{P}$ *strictly applicable to* E *in* $\langle F, A, \sigma, \gamma, \delta, \pi_i \rangle$ ***then***
      for all $(\phi \hookrightarrow \lambda) \in \max(E, \langle F, A, \sigma, \gamma, \delta, \pi_i \rangle, \mathcal{P})$ ***do***
        $\pi_{i+1} := \pi_{i+1} \cup \{add(E, \lambda, \phi)\}$*;*
      end for
    else
      $\pi_{i+1} := \pi_{i+1} \cup \{E\}$*;*
    end if
  end for
  i:=i+1;
until $\pi_i = \pi_{i-1}$*;*

Consider again the example of agent ag from section 2.2, its specification Δ_{ag} as given in section 3, and its generated goals $GoalExt(\Delta_{ag}, \mathcal{S})$ as explained in section 4. Let the coherence of a plan p be defined by the following resource constraint: p is coherent iff $Price(p) < \$2000$ where the function Price is recursively defined as follows: $Price(X;Y) = Price(X) + Price(Y)$. Assume that the price function is defined on primitive actions as follows:

$- Price(BuyKLMticketNY) = \$500$
$- Price(BuyBritishAirticketNY) = \$800$
$- Price(BuYKLMticketSF) = \$1000$
$- Price(BuyBritishAirticket - SF) = \$1300$
$- Price(ContactHotel1NY) = \$400$
$- Price(ContactHotel2NY) = \$500$
$- Price(ContactHotel1SF) = \$800$
$- Price(ContactHotel2SF) = \$1100$
$- Price(TransferMoney) = \$800$

For these generated goals, the procedure $PlanExt(\Delta, \mathcal{S})$ generate the following sets of plans:

$\{\{BuyKLMticketNY; StayHotel1NY\},$
$\{BuyKLMticketNY; StayHotel2NY\},$

$\{BuyBritishAirticketNY; StayHotel1NY\}$,
$\{BuyBritishAirticketNY; StayHotel2NY\}$,
$\{BuyKLMticketNY; StayHotel1NY; TransferMoney\}$,
$\{BuyKLMticketNY; StayHotel2NY; TransferMoney\}$,
$\{BuyBritishAirticketNY; StayHotel1NY; TransferMoney\}$,
$\{BuyKLMticketSF; StayHotel1SF\}\}$

5. Deliberation language

The observation, intention, and plan states of a BOID agent are determined by agent deliberation. In particular, the agent should deliberate to observe the environment after which the agent's observation state will be updated. Also, the agent decides to update its belief and goal states as these depend directly or indirectly on the agent's observation. The intention state of the agent is determined by a selection function which updates the intention state based on a subset of the agent's goal base. Finally, the plan state is determined by generating plans based on the agent's intention base and executing the plans. These decisions are specified by a deliberation program.

Definition 11 *A BOID deliberation program is an expression of the BOID deliberation language Θ defined as follows:*

- `Observe`, `UpdateBel`, `GenGoals`, `SelGoals`, `GenPlans`, `ExPlan` $\in \Theta$
- *if $\phi \in L$ and $\mu, \mu' \in \Theta$, then*
  - `if` ϕ `then` μ `else` $\mu' \in \Theta$
  - `while` ϕ `do` $\mu \in \Theta$

Definition 12 *(agent configuration) Let $\mathcal{S} = \langle F, A, \sigma, \gamma, \delta, \pi\rangle$ be a BOID state and s be a BOID deliberation program. A configuration of a BOID agent is a tuple $\langle s, \mathcal{S}\rangle$.*

A BOID agent configuration indicates the state of a BOID agent and the process that should transform the state. In this way, we consider the BOID state as the semantics of agent data (mental attitude) and the BOID deliberation program as the process that change the BOID state.

Definition 13 *Let P be a set of propositions, $env \subseteq P$ be the agent's environment at a certain moment, and $sense : P \rightarrow Pow(P)$ be the sense function such that $sense(env)$ indicates the data obtained by sensing the environment. Let also $UpdateSense : P \times P \rightarrow Pow(P)$ be the operator that updates the observation state F with the sensed data. Then, the semantics of the deliberation action* `Observe` *can be specified as follows:*

$$\frac{UpdateSense(sense(env), F) = F'}{\langle \texttt{Observe}, \langle F, A, \sigma, \gamma, \delta, \pi\rangle\rangle \rightarrow \langle \epsilon, \langle F', A, \sigma, \gamma, \delta, \pi\rangle\rangle}$$

A BOID agent may update its beliefs in a certain state. The belief update changes the belief state of the agent when agent has made new observations through which new belief rules can be applicable. The update of agent's beliefs is defined in terms of belief extension function which extends existing belief state.

Definition 14 *Let $\Delta = \langle \mathcal{B}, \mathcal{O}, \mathcal{I}, \mathcal{D}, \mathcal{P}, \rho\rangle$ be the specification of a BOID agent, and $\mathcal{S} = \langle F, A, \sigma, \gamma, \delta, \pi\rangle$ a state of the agent. Then, the semantics of the* `UpdateBel` *deliberation statement is defined by the following transition:*

$$\frac{BelExt(\Delta, \mathcal{S}) = \langle F, A', \sigma', \gamma, \delta, \pi\rangle}{\langle \texttt{UpdateBel}, \langle F, A, \sigma, \gamma, \delta, \pi\rangle\rangle \rightarrow \langle \epsilon, \langle F, A', \sigma', \gamma, \delta, \pi\rangle\rangle}$$

Similarly, an agent may generate its goals based on new observations and belief updates. The goal generation is defined in terms of the goal extension function which extends agent's goals.

Definition 15 *Let $\Delta = \langle \mathcal{B}, \mathcal{O}, \mathcal{I}, \mathcal{D}, \mathcal{P}, \rho\rangle$ be the specification of a BOID agent, and $\mathcal{S} = \langle F, A, \sigma, \gamma, \delta, \pi\rangle$ a state of the agent. Then, the semantics of the* `GenGoals` *deliberation statement is defined by the following transition:*

$$\frac{GoalExt(\Delta, \mathcal{S}) = \langle F, A', \sigma', \gamma', \delta, \pi\rangle}{\langle \texttt{GenGoals}, \langle F, A, \sigma, \gamma, \delta, \pi\rangle\rangle \rightarrow \langle \epsilon, \langle F, A', \sigma', \gamma', \delta, \pi\rangle\rangle}$$

Note that this transition updates the agent's belief state as well. This is because generated goals may trigger new belief rules the application of which extends the belief state of the agent. Moreover, `GenGoals;UpdateBel` = `GenGoals`.

In this paper, we assume that intentions are goals (obligations, intentions, and desires) that are chosen by the agent and to which the agent commits [6].

Definition 16 *Let sel_{goal} be a selection function which selects a goal (an extension) from the set of goals (set of extensions), and $UpdateInt$ be the intention update function. Then, the semantics of the* `SelGoals` *deliberation statement is defined by the following transition:*

$$\frac{UpdateInt(sel_{goal}(\gamma), \delta) = \delta'}{\langle \texttt{SelGoals}, \langle F, A, \sigma, \gamma, \delta, \pi\rangle\rangle \rightarrow \langle \epsilon, \langle F, A', \sigma, \gamma', \delta', \pi\rangle\rangle}$$

where $A' = A \backslash E$ such that $E \cap \{\mathbf{B}(\phi) \mid \phi \in L\} = sel_{goal}(\gamma)$ and $\gamma' = \{E' \cap \{\mathbf{O}(\phi), \mathbf{I}(\phi), \mathbf{D}(\phi) \mid \phi \in L\} \mid E' \in A'\}$.

This transition rule specifies the update of the intention state by a subset of goals. The selected subset of goals are removed from the agent's goals. One may think that the committed goals should remain a part of agent's goal state in which case they should not be removed from the goal state. This transition rule should then be modified appropriately.

In order to generate plans for an agent, the plan extension function is used to extend the agent's plan state.

Definition 17 *Let Δ be the specification of a BOID agent, and $\mathcal{S} = \langle F, A, \sigma, \gamma, \delta, \pi \rangle$ a state of the agent. Then, the semantics of the* `GenPlans` *deliberation statement is defined by the following transition:*

$$\frac{PlanExt(\Delta, \mathcal{S}) = \langle F, A, \sigma, \gamma, \delta, \pi' \rangle}{\langle \texttt{GenPlans}, \langle F, A, \sigma, \gamma, \delta, \pi \rangle \rangle \rightarrow \langle \epsilon, \langle F, A, \sigma, \gamma, \delta, \pi' \rangle \rangle}$$

This transition generate plans without removing the corresponding intentions from the intention state of the agent. One may argue that the planned intentions are not intentions anymore in which case this transition rule should be modified.

Plans can be selected from the agent's plan state and executed. We assume that an agent has a plan selection function. The execution of a plan depends on the structure of the plan. Therefore, we define several transition rules that specify the operational semantics of a type of plan.

Definition 18 *Let sel_{plan} be a function that selects one plan from a set of plans, $post(\alpha)$ be the post-condition or effect of action α, and $Update$ be an operator that updates the set of formulae from L with another set of formulae. Then, the semantics of execution of plans (primitive and composed) are defined as follows:*

- *Primitive actions are assumed to have effects on only the belief state of the agent. Alternatively, one may allow changes to other mental states of the agent as the effect of primitive actions.*

$$\frac{sel_{plan}(\pi) = \{\alpha\} \;\&\; Update(A, post(\alpha)) = A'}{\langle \texttt{ExPlan}, \langle F, A, \sigma, \gamma, \delta, \pi \rangle \rangle \rightarrow \langle \epsilon, \langle F, A', \sigma', \gamma, \delta, \pi' \rangle \rangle}$$

  where $\sigma' = \{E' \cap \{\mathbf{B}(\phi) \mid \phi \in L\} \mid E' \in A'\}$ and $\pi' = \pi \backslash \{\{\alpha\}\}$.

- *Test actions have no effect on the agent's state. It can only blocks the execution if the test does not succeed.*

$$\frac{sel_{plan}(\pi) = \{\phi?\} \;\&\; \langle F, A, \sigma, \gamma, \delta, \pi \rangle \models \phi}{\langle \texttt{ExPlan}, \langle F, A, \sigma, \gamma, \delta, \pi \rangle \rangle \rightarrow \langle \epsilon, \langle F, A, \sigma, \gamma, \delta, \pi' \rangle \rangle}$$

  where $\pi' = \pi \backslash \{\{\phi?\}\}$.

- *The execution of partial plans (goal formula) is accomplished by planning the goal, i.e. by applying a planning rule. Note that the agent continues with the execution of the plan.*

$$\frac{sel_{plan}(\pi) = \{\psi\} \;\&\; \psi \hookrightarrow \lambda \in \mathcal{P}}{\langle \texttt{ExPlan}, \langle F, A, \sigma, \gamma, \delta, \pi \rangle \rangle \rightarrow \langle \texttt{ExPlan}, \langle F, A, \sigma, \gamma, \delta, \pi' \rangle \rangle}$$

  where $\pi' = (\pi \backslash \{\{\psi\}\}) \cup \{\{\lambda\}\}$.

- *The execution of (sequence) plan $\alpha; \lambda$ is accomplished by executing α first and λ thereafter.*

$$\frac{\begin{array}{c} sel_{plan}(\pi) = \{\alpha; \lambda\} \;\& \\ \langle \texttt{ExPlan}, \langle F, A, \sigma, \gamma, \delta, \pi \cup \{\{\alpha\}\} \rangle \rangle \rightarrow \\ \langle \texttt{ExPlan}, \langle F, A', \sigma', \gamma, \delta, \pi \rangle \rangle \end{array}}{\begin{array}{c} \langle \texttt{ExPlan}, \langle F, A, \sigma, \gamma, \delta, \pi \cup \{\{\alpha; \lambda\}\} \rangle \rangle \rightarrow \\ \langle \texttt{ExPlan}, \langle F, A', \sigma', \gamma, \delta, \pi \cup \{\{\lambda\}\} \rangle \rangle \end{array}}$$

- *The execution of if-then-else plan depends on the satisfiability of the condition. If condition of the if-then-else plan is derivable from agent's state, then the execution of the plan continues with the 'then' part of the plan. Let $s = (\ \texttt{if}\ \phi\ \texttt{then}\ \lambda\ \texttt{else}\ \lambda')$, then,*

$$\frac{sel_{plan}(\pi) = \{s\} \;\&\; \langle F, A, \sigma, \gamma, \delta, \pi \rangle \models \phi}{\begin{array}{c} \langle \texttt{ExPlan}, \langle F, A, \sigma, \gamma, \delta, \pi \rangle \rangle \rightarrow \\ \langle \texttt{ExPlan}, \langle F, A, \sigma, \gamma, \delta, \pi' \rangle \rangle \end{array}}$$

  where $\pi' = (\pi \backslash \{\{s\}\}) \cup \{\{\lambda\}\}$.

- *If condition of the if-then-else plan is not derivable from the beliefbase, the execution of the plan continues with the 'else' part of the plan. Let $s = (\ \texttt{if}\ \phi\ \texttt{then}\ \lambda\ \texttt{else}\ \lambda')$, then,*

$$\frac{sel_{plan}(\pi) = \{s\} \;\&\; \langle F, A, \sigma, \gamma, \delta, \pi \rangle \not\models \phi}{\begin{array}{c} \langle \texttt{ExPlan}, \langle F, A, \sigma, \gamma, \delta, \pi \rangle \rangle \rightarrow \\ \langle \texttt{ExPlan}, \langle F, A, \sigma, \gamma, \delta, \pi' \rangle \rangle \end{array}}$$

  where $\pi' = (\pi \backslash \{\{s\}\}) \cup \{\{\lambda'\}\}$.

- *The execution of while-do plan depends on the satisfiability of the condition. If the condition is derivable from the beliefbase, the plan is prefixed with its body. Let $s = (\ \texttt{while}\ \phi\ \texttt{do}\ \lambda)$, then,*

$$\frac{sel_{plan}(\pi) = \{s\} \;\&\; \langle F, A, \sigma, \gamma, \delta, \pi \rangle \models \phi}{\begin{array}{c} \langle \texttt{ExPlan}, \langle F, A, \sigma, \gamma, \delta, \pi \rangle \rangle \rightarrow \\ \langle \texttt{ExPlan}, \langle F, A, \sigma, \gamma, \delta, \pi' \rangle \rangle \end{array}}$$

  where $\pi' = (\pi \backslash \{\{s\}\}) \cup \{\{\lambda; s\}\}$.

- *If the condition of the while-do plan is not derivable from the beliefbase, the plan is removed. Let $s = (\ \texttt{while}\ \phi\ \texttt{do}\ \lambda)$, then,*

$$\frac{sel_{plan}(\pi) = \{s\} \;\&\; \langle F, A, \sigma, \gamma, \delta, \pi \rangle \not\models \phi}{\langle \texttt{ExPlan}, \langle F, A, \sigma, \gamma, \delta, \pi \rangle \rangle \rightarrow \langle \epsilon, \langle F, A, \sigma, \gamma, \delta, \pi' \rangle \rangle}$$

  where $\pi' = (\pi \backslash \{\{s\}\})$.

The above definitions specified the semantics of the primitive deliberation statement. The semantics of the composed deliberation statements can now be defined in standard way as follows:

Definition 19 *Let $\phi \in L$ and $\mu, \mu' \in \Theta$ be deliberation programs. The semantics of if-then-else and while-do deliberation programs are based on the agent's state.*

$$\frac{\langle F, A, \sigma, \gamma, \delta, \pi \rangle \models \phi}{\begin{array}{c} \langle\ \texttt{if}\ \phi\ \texttt{then}\ \mu\ \texttt{else}\ \mu', \langle F, A, \sigma, \gamma, \delta, \pi \rangle \rangle \rightarrow \\ \langle \mu, \langle F, A, \sigma, \gamma, \delta, \pi \rangle \rangle \end{array}}$$

$$\frac{\langle F, A, \sigma, \gamma, \delta, \pi\rangle \not\models \phi}{\langle \texttt{ if } \phi \texttt{ then } \mu \texttt{ else } \mu', \langle F, A, \sigma, \gamma, \delta, \pi\rangle\rangle \rightarrow \langle \mu', \langle F, A, \sigma, \gamma, \delta, \pi\rangle\rangle}$$

$$\frac{\langle F, A, \sigma, \gamma, \delta, \pi\rangle \models \phi}{\langle \texttt{while } \phi \texttt{ do } \mu, \langle F, A, \sigma, \gamma, \delta, \pi\rangle\rangle \rightarrow \langle \mu; \texttt{while } \phi \texttt{ do } \mu, \langle F, A, \sigma, \gamma, \delta, \pi\rangle\rangle}$$

$$\frac{\langle F, A, \sigma, \gamma, \delta, \pi\rangle \not\models \phi}{\langle \texttt{while } \phi \texttt{ do } \mu, \langle F, A, \sigma, \gamma, \delta, \pi\rangle\rangle \rightarrow \langle \epsilon, \langle F, A, \sigma, \gamma, \delta, \pi\rangle\rangle}$$

The above transitions allow the agent's deliberation to depend on the agent's state. In this case, one can program an agent whose deliberation depends on its senses, beliefs, goals, intentions, etc. For example, one may implement an agent in such a way that when the agent believes that resources are running out, then the agent will not generate a plan anymore and try to execute as many of its plans as possible.

6. Concluding remarks

In most traditional agent programming languages, such as agentspeak [12] or the original 3APL agent programming language proposed by Hindriks et al [9], the agent programmer can program initial mental states like the beliefs, desires, goals, and plans. Only in a few of them the agent programmer can implement the deliberation process, such as in the most recent version of 3APL [7]. However, the deliberation instructions in 3APL are limited to goal selection, applying planning rules to selected goals, and executing partial plans. There is no programming instruction available to generate the sets of possible goals from which one can be selected. One reason for this is related to the fact that the goals are assumed to be compatible such that there always exists only one set of goals. There is also no programming instruction to generate possible sets of plans, such that it is not possible to consider and evaluate plans before selecting them. In this paper we extend 3APL in the following ways:

1. We introduce so-called generate and select deliberation, with programming primitives such as observe, generate goal sets, select goal sets, generate plans, select plans, and execute plans.
2. The programmer can determine the order of these primitives in the deliberation cycle, which may depend on the context. For example, in some contexts the agent may spend time on deliberating about new goals, whereas in other contexts the agent should only plan, or only execute actions. The context is here the agent's mental state, typically its beliefs. For example, when the agent believes to be in a state of emergency, he should not think but act. This allows the programmer to balance these primitives, and in particular the goal generation and planning process.
3. Moreover, the agent programmer can define subroutines used in the generate and select procedures. For example, the programmer can define how many new goals or new plans must be generated, and he can add routines that determine whether a goal or a plan is coherent or not.

Alternatively, from the perspective of the BOID architecture, which is based on propositional rule based logic, we have defined an operational semantics for BOID agents defined in [4], and we have extended BOID agent reasoning with modal operators.

References

[1] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[2] M. Bratman. *Intention, plans, and practical reason*. Harvard University Press, Cambridge Mass, 1987.

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

[4] J. Broersen, M. Dastani, J. Hulstijn, and L. van der Torre. Goal generation in the BOID architecture. *Cognitive Science Quarterly*, 2(3-4):428–447, 2002.

[5] R. Brooks. How to build complete creatures rather than isolated cognitive simulators. In K. VanLehn, editor, *Architectures for Intelligence*, pages 225–239. Lawrence Erlbaum Assosiates, Hillsdale, NJ, 1991.

[6] P. Cohen and H. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

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

[8] M. Dastani, F. Dignum, and J.-J. Meyer. Autonomy and agent deliberation. In *Proceedings of The First International Workshop on Computatinal Autonomy - Potential, Risks, Solutions (Autonomous 2003)*, Melbourne, Australia, 2003.

[9] K. V. Hindriks, F. S. D. Boer, W. V. der Hoek, and J.-J. C. Meyer. Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.

[10] A. Rao and M. Georgeff. Modeling rational agents within a BDI architecture. In *Proceedings of the KR91*, 1991.

[11] A. Rao and M. Georgeff. BDI agents: From theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95)*, pages 312–319, 1995.

[12] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In W. van der Velde and J. Perram, editors, *Agents Breaking Away (LNAI 1038)*, pages 42–55. Springer-Verlag, 1996.