The BOID Architecture

Conflicts Between Beliefs, Obligations, Intentions and Desires

Jan Broersen Mehdi Dastani Joris Hulstijn Zisheng Huang Leendert van der Torre
Division of Mathematics and Computer Science
Vrije Universiteit
de Boelelaan 1081a
1081 HV Amsterdam
http://www.cs.vu.nl/~boid
{broersen,mehdi,joris,huang,torre}@cs.vu.nl

ABSTRACT

In this paper we introduce the so-called Beliefs-Obligations-Intentions-Desires or BOID architecture. It contains feedback loops to consider all effects of actions before committing to them, and mechanisms to resolve conflicts between the outputs of its four components. Agent types such as realistic or social agents correspond to specific types of conflict resolution embedded in the BOID architecture.

1. INTRODUCTION

Various competing decision models for autonomous agents have been proposed, and it is still unclear which type of model should be used in which type of application. For example, some decision models are based on goal-based planning or on variants of decision theory like qualitative decision theory [15, 3], other models are based on cognitive models like belief-desire-intention models [7, 16], and yet other models are based on social concepts like obligations and norms [10, 23, 22], as in deontic action programs [12]. Typically, the decision model is based on an attempt to reach goals, satisfy desires, fulfill obligations etc. Here we consider decision models for an agent that is overloaded with input, and typically lives in a complex and noisy environment. His main problem is not to find a way to reach his goals, satisfy his desires or fulfill his obligations, but which of the desires and obligations he will follow given his beliefs and intentions. That is, his main problem is to resolve the conflicts among his attitudes.

In this paper we propose the Beliefs-Obligations-Intentions-Desires or BOID architecture, an agent architecture that contains at least four components. As these components output beliefs, obligations, intentions and desires only for certain inputs, they represent *conditional* informational and motivational attitudes. Conflicts between these outputs are either resolved by the architecture's control loop or by a separate selection component that outputs new intentions.

- Agent types are represented by control loops. In a realistic agent beliefs override obligations, intentions or desires; in a single-minded or stable agent intentions override desires and obligations; in an open-minded or unstable agent desires and obligations override intentions; in selfish agents desires override obligations and in social agents obligations override desires.
- For other conflicts so-called extensions are constructed, and one extension is selected. This idea is adopted from Thomason's BDP logic [21], which is again based on Reiter's default logic [19]. To represent all effects of actions before committing to them the architecture is based on feedback loops, as is explained in detail later in this paper using the Al-Bob-Chris example of Dignum *et.al.* [11].

In the implementation of the BOID architecture discussed in this paper the content of the informational and motivational attitudes is represented by propositional formulas. In the simplest BOID the four components have as input a set of formulas – called an extension – and as output another extension, and in the full BOID the input and output are sets of extensions. To resolve the second type of conflict we add another component to select an extension. The output of this component, the new intentions, is the input for a planning component.

In this paper we focus on the implementation of the BOID architecture. The BOID logic is discussed in more detail elsewhere [5]. We focus on a single autonomous agent for whom other agents are only important in as far as they are represented implicitly in norms and social commitments. Further multi-agent extensions, such as for example conventions to coordinate joint plans and trust to cooperate between competitive agents, are outside the scope of this paper. However, we are aware that use of the name 'boid' compels us to deal with large flocks of agents in future research.

The layout of the paper is as follows. In Section 2 different types of conflicts are discussed and a classification of agent types is introduced. In Section 3 the BOID architecture, logic, and a control loop are introduced. In Section 4 examples from Section 2 are formalized in the BOID architecture and implementation details are discussed.

AGENTS'01, May 28-June 1, 2001, Montréal, Quebec, Canada.

2. BELIEFS, OBLIGATIONS, INTENTIONS AND DESIRES

Reasoning about beliefs, obligations, intentions and desires has been discussed in practical reasoning in philosophy [24, 4], and its formalization to build intelligent autonomous agents has more recently been discussed in qualitative decision making in artificial intelligence [11, 12, 18, 21]. On closer inspection each of these four concepts consists of related (though often quite distinct) concepts, for example respectively knowledge and defaults, prohibitions and permissions, commitments and plans, wishes and wants. All these concepts are grouped into these four classes due to their role in the decision making process: beliefs are informational attitudes – how the world is expected to be – obligations and desires are the external and internal motivational attitudes, and intentions are the results of decision making.

In this section we focus on the interaction between these classes. In particular, we discuss fifteen types of conflicts which can occur between these four concepts, we discuss how different types of agent resolve these conflicts in different ways, and we discuss some of the problems related to conflict resolution. This section provides the necessary background to understand the choices made in the BOID architecture, which is presented in the Section 3.

2.1 Four concepts

Beliefs and desires are informational and motivational attitudes which can be related to two structures in all other models of decision making. For example, they can be related to respectively probabilities and utilities in classical decision theory. Our interpretation of beliefs and desires is inspired by the BD logic of Thomason [21], though other interpretations can be given as well.

Intentions are introduced to relate previous decisions to new decisions. In the philosophical literature it has been argued, among others by Bratman [4], that intentions deserve a special status besides beliefs and desires: they cannot be reduced to them. This special status is the central focus of the computational BDI approaches of Cohen and Levesque [7], and of Rao and Georgeff [16, 17, 18]. Prior intentions are the actions the agent has committed to in his previous decisions. The incorporation of intentions makes the agent's behavior more stable [16, 17] and makes it possible to take bounded reasoning into account [4].

Obligations are the most controversial component in our architecture. One reason to introduce this component is to incorporate obligations, norms and commitments of social agents and social rationality. However, there is another argument taken from research in deontic logic and computer science [14]. The question has been raised why norms are usually not implemented explicitly in computer systems. An easy answer is that computer programs already model 'ideal' behavior. They must never violate the rules, just as they must never fail. This objection can be countered by Dignum's argument [10] that obligations can be violated, because agents are autonomous. In a typical example, an agent has a desire to do otherwise and the desire is stronger than the obligation. Even social agents can violate their obligations if they intended earlier to do otherwise and are not open-minded enough to reconsider this intention. Examples of such cases are given below. Finally, in order to deal with conflicts among norms, and agent must be able to drop some obligations in favor of others.

2.2 Possible Conflicts

One of the main tasks of deliberative agents is to solve possible conflicts among informational and motivational attitudes. In this subsection we list fifteen different types of conflicts that may arise either within each class or between classes. Dependent on the exact interpretation of these classes, some of the conflict types may be more interesting or important than others. We distinguish two general types of conflicts: internal and external conflicts. *Internal conflicts* are caused within each component while *external conflicts* are caused between them. Internal conflicts can be distinguished into four unary subtypes (B ; O ; I ; D).

B conflict: [21] *I have a reason to believe the porch light is off, because I asked my daughter to turn it off. I have a reason to believe the porch light is on, because the last time I saw it, it was on.*

O conflict: *It is obligatory to be honest. It is obligatory to be polite. If I am honest about it, it will be impolite.*

I conflict: [2] *I intend to finish the paper on Sunday. I intend to go to the beach on Saturday. If I go to the beach on Saturday, I cannot finish the paper on Sunday.*

D conflict: [8] *I desire to smoke. I desire to be healthy. However, smoking endangers my health.*

External conflicts can be distinguished into six binary conflict subtypes (BO ; BI ; BD ; OI; OD ; ID), and four ternary conflict types (BOI; BOD; BID; OID) and one quadruplicate conflict type (BOID).

BO conflict: *It is obligatory to see my mother-in-law this weekend. But I think I have no time to go.*

BI conflict: *I have a plan to see my mother-in-law this weekend. However, I find it is impossible for me now, for my car is broken.*

BD conflict: [21] *I'd like to take a long vacation. I'd need to get time off from work to take a long vacation. But I can't get time off from work.*

OI conflict: *It is obligatory to see my mother-in-law this weekend. But, I already have a plan to go to a conference. If I go to the conference, I cannot go to see my mother-in-law.*

OD conflict: *It is obligatory not to smoke in a non-smoking area. I desire to smoke in my office. However, my office is a non-smoking area.*

ID conflict: [21] *I'd like to take a nap. But I intend to catch a plane.*

BOI conflict: *If I smoke, I should smoke in a smoking area. I intend to smoke. However, I know that it is a non-smoking area here.*

BOD conflict: *If I smoke, I should smoke in a smoking area. I desire to smoke. However, I know that it is a non-smoking area here.*

BID conflict: *I intend to go to a conference. I desire that the travel cost is not too expensive. I know that if I go to the conference, then the travel cost would be very expensive.*

OID conflict: *I intend to go to a conference. I desire to stay in a luxury hotel. However, it is obligatory for me that if I go to the conference, I should not stay in a luxury hotel.*

BOID conflict: *I intend to go to a conference. It is obligatory for me not to spend too much money for the conference. Namely, either I should pay for a cheap flight ticket and stay in a better hotel, or I should pay for an expensive flight ticket and stay in a budget hotel. I desire to stay in a better hotel. But, I know that the secretary has booked an expensive flight ticket for me.*

In the following subsection we structure the resolution of conflicts for these types. A classification of conflict resolution types is introduced and discussed. It is argued that these types of conflict resolutions correspond with what in agent theories is called *agent types.* Some well-known agent types are for example *realistic, selfish, social, simple-minded*, and *open-minded* agents.

2.3 Conflict resolution and agent types

A conflict resolution type is an order of overruling. Given four components, there are 24 possible orders of overruling. In this paper, we only consider those orders according to which the belief component overrules any motivational attitude component. This reduces the number of possible overruling orders to six. Some examples of conflict resolution with beliefs are given below.

- A conflict between a belief and a prior intention means that an intended action can no longer be executed due to the changing environment. Beliefs therefore overrule the prior intention, which is retracted. Any derived consequences of this prior intention are retracted too. Of course, one may allow prior intentions to overrule beliefs, but this results in unrealistic behavior.
- Analogously, a conflict between a belief and obligation or desire means that a violation has occurred. As observed by Thomason [21], the beliefs must override the desires or otherwise there is wishful thinking; the same argument applies to obligations.

Moreover, a conflict between a prior intention and an obligation or desire means that you now should or want to do something else than you intended before. Here prior intentions override the latter because it is exactly this property for which intentions have been introduced: to bring stability. Only in a call for intention reconsideration such conflicts may be resolved otherwise. For example, if I intend to go to the cinema but I am obliged to visit my mother, then I go to the cinema unless I reconsider my intentions.

Using the order of string letters as the overruling order, these six ways of resolving conflicts can be represented as BOID, BODI, BDIO, BDOI, BIOD, and BIDO. Note that we overloaded the name BOID in this way, because it becomes a certain type of agent as well as the general name for the agent architecture.

Realistic. The six conflict resolution types (agent types) in which beliefs override all other components are called *realistic*.

Simple-minded. BIDO and BIOD are called *simple-minded* or *stable*, because prior intentions overrule desires and obligations.

Selfish. BDIO and BDOI are called *selfish*, because desires overrule obligations.

Social. BIOD, BOID and BODI are called *social*, because obligations overrule desires.

Other classifications are also possible. For example, we may call agents super-selfish or super-social if they are respectively selfish and social but not simple-minded. This means that super-selfish and super-social agents start with respectively BD and BO. Moreover, we can have partial prioritization constraints. Examples of those are discussed in Section 3.3. Summarizing, conflicts can be resolved according to a priority ordering.

2.4 Minimality + conditionals = complications

There are several complications to further specify and implement the conflict types and their associated agent types. It may seem that we can use one of the many approaches to conflict resolution developed in other areas of artificial intelligence like for example diagnosis [20], default reasoning or fusion of knowledge and databases. However, there is a problem. Regardless of the exact definition of a conflict, in these approaches a conflict is always defined as a *minimal* set, in the sense that if two sets are conflicting then one of the sets cannot be a strict subset of the other one. Whereas minimal sets may be the obvious choice in diagnosis and other applications, it is problematic in decision making with conditionals.

An example has been given by Dignum *et.al.* [11], who discuss an extension of the BDI architecture with obligations. In this example, there is a guy called Al who has an obligation to perform a task for Bob and another incompatible obligation to perform a task for Chris. Moreover, Al has the norm that he should tell Bob if he does not intend to meet this obligation. The problem discussed in the paper is that the existence of the norm should affect Al's decision on whether to intend to fulfill his obligation:

> "Consider Al's obligation above, until he actually commits to not meeting his obligation to Bob, the need to tell Bob does not exist, yet the *potential* for it may have a significant impact on his decision on whether to do the task for Bob. For example, imagine that the task is trivial (i.e., the direct consequences of not doing the task are small), but the social consequences of not informing Bob are very high (i.e., Al is perceived as unreliable)." [11, p.115]

The point is thus that to resolve the conflict we cannot restrict ourselves to the minimal set (the two obligations), but we have to consider the whole set. In general, agents should consider the effects of actions before they commit to it. This is the reason why in the BOID architecture discussed next complete extensions are constructed before one is selected, instead of solving a conflict as one is encountered.

3. AGENT ARCHITECTURE

In this section we discuss the BOID architecture. We first discuss the case in which all conflicts can be resolved by the agent type, like the examples in Section 2.2. Thus it needs to build only a single set of formulas as output: a *single extension* (Section 3.2). After that, we discuss the full BOID which also covers more complex cases like the example in Section 2.4. This second architecture calculates *multiple extensions* as output of the components (Section 3.3). We introduce an additional component that selects one final extension, which represents the agent's intentions.

3.1 Components

In general, an agent can be seen as a black box with observations as input and intended actions as output, which are related to the environment by detectors and effectors. In the BOID architecture these attitudes are mapped to four components within the agent architecture, in the sense that each component outputs one of the attitudes. The components associated with an attitude can be implemented in a variety of different ways. For example, the *Beliefs* component may maximize cross entropy or apply AGM belief revision [1], and its output may be a probability distribution, a set of them, plausibility measures, a belief set, etc. Moreover, the *Desires* component may be based on a quantitative utilitarian model and maximize expected utility to determine goals. The *Obligations* and *Intentions* components reason with personal as well as social obligations and commitments to select goals and plans to reach selected goals.

In the BOID architecture discussed in this paper, the behavior of each component is specified by propositional logical formulas, often in the form of defeasible rules. The input and output of the components is represented by sets of logical formulas, closed under logical consequence. Following Thomason [21] these are called *extensions*. We distinguish between the agent's static architecture and its dynamic behavior. In our approach, the former only concerns the agent whereas the latter concerns the agent with its environment.

3.2 Single Extension BOID

We start with a BOID architecture that builds only one extension. The logic that specifies the behavior of the architecture is parameterized with an ordering function ρ to resolve conflicts. It constraints the order in which derivation steps for different components are undertaken and characterizes the type of BOID. We first discuss the BOID logic, then the agent types and finally the dynamic control loop that determines how the BOID interacts with its environment.

3.2.1 Logic or calculation scheme

Each calculation starts with a set of observations W. Unlike normal beliefs, which may have a default character, observations can not be overridden. We assume initial sets of defeasible rules for the other components: B,O, I^-, D. We write I^- to emphasize that the set I contains *prior* intentions.

We first define an ordering function ρ on rules that represents the type of agent. In case of multiple applicable rules, the one with the lowest ρ value is applied. In this version ρ is *complete*: it assigns a unique value to each rule. Therefore it not only resolves conflicts between components, but also among rules within components. Given ρ, the calculation for building extensions can now be defined as follows.

Definition 1 (BOID Calculation Scheme). *Let L be a propositional language, a tuple $\Delta = \langle W, B, O, I^-, D\rangle$ a BOID theory with W a subset of L and B, O, I^- and D sets of ordered pairs of L written as $\alpha \hookrightarrow w$, and ρ be a function from $B \cup O \cup I^- \cup D$ to the integers.*
We say that a rule $(\alpha \hookrightarrow w)$ is applicable *to an extension E, iff $\alpha \in E$ and $\neg w \notin E$.*
Define

$$E_0 = W \qquad \text{and for } i \geq 0$$

$$E_{i+1} = Th_L(E_i \cup \{w \mid (\alpha \hookrightarrow w) \in B \cup O \cup I^- \cup D \text{ and } (\alpha \hookrightarrow w) \text{ is applicable to } E_i \text{ and } \nexists(\beta \hookrightarrow v) \in B \cup O \cup I^- \cup D \text{ applicable to } E_i \text{ such that } \rho(\beta \hookrightarrow v) < \rho(\alpha \hookrightarrow w)\ \}\ ).$$

Then $E \subseteq L$ is an extension for Δ iff $E = \cup_{i=0}^{\infty} E_i$.

In practice not the whole extension is calculated in the architecture (since this may be infinite), but only the set of outputs w or the set of rules $\alpha \hookrightarrow w$ that can be calculated before the agent runs out of resources.

3.2.2 Agent types

In the BOID architecture, we start with the observations and calculate a belief extension by iteratively applying belief rules. When no belief rule is applicable anymore, based on the agent type (i.e. conflict resolution type), either the O, the I^-, or the D component is chosen from which one applicable rule is selected and applied. When a rule from a chosen component is applied successfully, the belief component is attended again and belief rules are applied. If there is no rule from the chosen component applicable, then again based on the agent type the next component is chosen. If there is no rule from any of the components applicable, then the process terminates – a fixed point is reached – and one extension is calculated. For the calculation scheme in Definition 1 this approach means that ρ is constructed as follows.

Definition 2 (Agent Types). *Let B, O, I^-, and D be the mutually exclusive sets of rules for beliefs, obligations, prior intentions, and desires, respectively. Let also X and Y be any of these sets. An* agent type *is defined as a function $\rho : B \cup O \cup I^- \cup D \to \mathbf{N}$ that assigns a unique integer to each rule from $B \cup O \cup I^- \cup D$ such that for $X \neq Y$:*

$$\forall r_x \in X\ \forall r_y \in Y\ \ \rho(r_x) < \rho(r_y)\ \ \vee$$
$$\forall r_x \in X\ \forall r_y \in Y\ \ \rho(r_y) < \rho(r_x).$$

Note that ρ assigns unique values to the rules of all components such that the values of all rules from one component are either smaller or greater than the values of all rules from another component. The agent types of section 2 can now be characterized as follows.

social simple-minded or stable agent
$\rho(r_b) < \rho(r_{i^-}) < \rho(r_o) < \rho(r_d)$

selfish simple-minded agent
$\rho(r_b) < \rho(r_{i^-}) < \rho(r_d) < \rho(r_o)$

social open-minded agent
$\rho(r_b) < \rho(r_o) < \rho(r_d) < \rho(r_{i^-})$

selfish open-minded agent
$\rho(r_b) < \rho(r_d) < \rho(r_o) < \rho(r_{i^-})$

3.2.3 Specialized architectures

An agent architecture specifies the components of an agent, how they are related, and how the information flows around. The combination of the calculation scheme with an agent type induces a certain agent architecture. For example, consider the social simple-minded agent type, with ρ defined as above. This agent type induces the architecture illustrated in Figure 1. It should be interpreted as follows. Each component receives an input extension and generates an output extension. If the input and output extensions are identical (i.e. no new rules can be applied), the output extension flows to the next component, otherwise it flows back through the feedback loop. The initial extension is based on a set of observations, which can be empty. Then, belief rules are applied iteratively, indicated by the feedback loop around the B component. If no more belief rules are applicable, then the calculated extension is sent to the I^- component. If possible, one prior intention is applied and the extension is sent back to the B component via the feedback loop from I^- to B; otherwise the extension goes to the O component, etc.

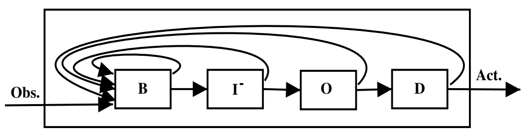

Figure 1: social simple minded

3.2.4 General architecture

As we have seen, all six realistic agent types share one characteristic: there are indefeasible observations, and the belief component overrules all other components. The order in which the other components are applied depends on the ρ parameter. That means that, when ρ is considered as a parameter of the underlying logic of extension calculations, a general agent architecture can be proposed for all agent types.

In addition to decision making, also the *planning process* plays an important role. Planning is needed to decide which actions should be performed in order to achieve the intentions represented by the calculated extension. For this reason, an additional component P is added to determine which actions should be performed. The input to the planning component is an extension; the output is set of actions scheduled to be performed. The resulting general agent architecture is illustrated in Figure 2. The architecture should be interpreted as above: if the output extension of a component differs from the input extension, it flows back through the feedback loop; otherwise it flows to one of accessible components determined by the ρ function.

3.2.5 Control loop

Consider a BOID agent, configured by a certain ρ, in a dynamic environment. It receives input from the environment, calculate an extension, decides which actions should be performed, updates all components, and starts observing the environment again. The agent type ρ together with this order of processes define the control loop for the BOID architecture, which determines the behavior of the agent in

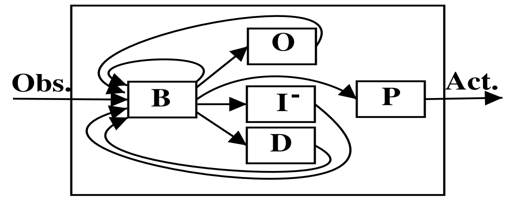

Figure 2: BOID Architecture

dynamic environment. This control loop can be written as follows:

set ρ;
repeat
E := calculate_extension(Observations,ρ);
plan_extension(E);
update(B, O, I^-, D)
until forever

Note that the extension calculation which is a part of the control loop is itself a complex process which is explained in Definition 1.

3.3 Multiple Extension BOID

To account for the general situation, in this subsection we propose the multiple extension BOID architecture, which calculates a set of extensions instead on one single extension. There can be two reasons for multiple extensions. First, there can be larger sets than the one discussed in Section 2.2, like for example the examples discussed in Section 2.4. Second, the examples discussed in Section 2.2 can lead to multiple extensions if the agent type is less specific than the ones discussed thus far.

3.3.1 More agent types

Let $r_x \in X$ where X is either B, O, I^-, or D. Then, based on Definition 2 some agent types that are introduced in the previous section can be defined as a ρ function with certain properties.

simple-minded agent
$\rho(r_b) < \rho(r_{i^-}) < \rho(r_o)$ and $\rho(r_b) < \rho(r_{i^-}) < \rho(r_d)$

super-social agent
$\rho(r_b) < \rho(r_o) < \rho(r_{i^-})$ and $\rho(r_b) < \rho(r_o) < \rho(r_d)$

super-selfish agent
$\rho(r_b) < \rho(r_d) < \rho(r_{i^-})$ and $\rho(r_b) < \rho(r_d) < \rho(r_o)$

The super-selfish agent type, for which the ρ function is defined above, induces the agent architecture illustrated in Figure 3. In this architecture, the extension generated by the D component goes back to the B component by the feedback loop if one desire rule is successfully applied. Otherwise, it goes either to the I^- component or to the O component. This non-determinism indicates a choice of the super-selfish agent. For example, consider the OI conflict in which it is obligatory to see my mother-in-law this weekend, but I already have an intention to go to a conference. A selfish agent cannot resolve this conflict automatically by his type, and has to decide in some other way.

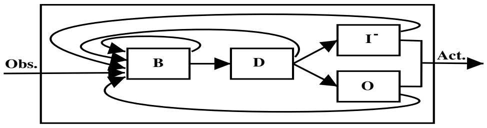

Figure 3: super-selfish

3.3.2 Calculation scheme

Of course, in order to calculate multiple extensions, the BOID logic, the ρ function, and the control loop need to be adapted as well. For example, the ρ function will not assign unique numbers to the rules of one components such that more than one rule can be applied at a certain stage of extension calculation.

Definition 3 (BOID Calculation Scheme). *Let $\Delta = \langle W, B, O, I^-, D\rangle$ be a BOID theory and ρ be an agent type function.*
Define
$S_0 = \{W\}$, *and for* $i \geq 0$
$S_{i+1} = \{ Th_L(E \cup \{w\}) \mid E \in S_i,$
$(\alpha \hookrightarrow w) \in B \cup O \cup I^- \cup D$ *and*
$(\alpha \hookrightarrow w)$ *is applicable to* E *and*
$\nexists(\beta \hookrightarrow v) \in B \cup O \cup I^- \cup D$ *applicable to* E
such that $\rho(\beta \hookrightarrow v) < \rho(\alpha \hookrightarrow w)$,
if such $(\alpha \hookrightarrow w)$ *exist, otherwise* $w = \top$ }) }.
Then $E \subseteq L$ *is an extension for* Δ *iff* $S = \cup_{i=0}^{\infty} S_i$.

3.3.3 Extension selection

An additional component, called *new intention component* (I^+) is added resulting in the BOID architecture illustrated in Figure 4. This component selects one extension from the calculated set of extensions and sends it to the planning component. The OI conflict above can then be modeled by an interaction between the I^+ and the planning components. In fact, the I^+ component is assumed to impose an ordering on the input extensions (based on an extension selection strategy) such that it can select and send the best extension to the planning component. If for any reason the selected extension cannot be translated to a feasible plan, the I^+ component sends the next best extension to the planning component.

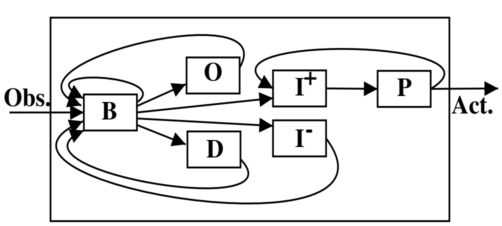

Figure 4: Multiple Extension BOID

The update of the component in the control loop needs to be modified as well. A specific update is based on the fact that the selected extension indicates new intentions, which forms the prior intentions for the next round of deliberation. Therefore, beside updating all components, the I^- component is updated based on the selected extension.

As noted the I^+ component imposes an ordering on input extensions based on an extension selection strategy. Here are some options, which all have their drawbacks. But remember, the agent has to act so even if the extension selection is problematic, he has to chose something.

1. Select an extension randomly;
2. First select the beliefs, then select the desires;
3. Choose desires such that it minimizes the conflicts between beliefs;
4. Select maximal or minimal extensions;
5. Select extensions with more beliefs, or more desires, etc;
6. Select extensions that have larger intersection with most other extensions.

There are two more important issues which have to be addressed in a study of decision-making as choosing extensions. First, apart from choosing an extension we can also decide a conflict between beliefs by simply making an observation, and check in the world which belief is true. This leads us to the well-known problems of planning with partial observations. In general, we can use an oracle to test whether a proposition is in or out of the extension. Second, in practice we have to add some mathematical structure to the rules, with for example priorities as in prioritized rule logic. We now have much more complexity due to our components! Another option is to go more decision-theoretic and associate losses and gains with the rules. This is by itself a line of research with many interesting problems [9]. These issues of extension selection are left for further research.

4. IMPLEMENTATION AND EXAMPLES

The idea behind the BOID architecture is very general; components can be implemented by any mechanism that can produce a certain input-output behavior. In the description above we assumed that components are described by simple production rules. However, this assumption is not crucial.

4.1 Prolog Prototype

Nevertheless we found it useful to implement a prototype of the BOID in Prolog, along with the examples of section 2. In this way one can try out different agent types, and see if their behavior on the examples comes out as expected. The source code can be found at http://www.cs.vu.nl/~boid/. We made the following implementation choices.

1. Components are implemented by production rules of the form x_rule(Index, A —> W), where is x is either B, D, I or O, and where Index is a variable that indicates the particular example or situation that is modeled. A and W are both formulas of propositional logic, where the consequent W does not contain any disjunctions. Rules with disjunctions in the consequent can be replaced by two rules with the same antecedent; one for each disjunct. Unconditional beliefs, desires, intentions or obligations are represented by a rule with $\top$ or true in the antecedent.

2. Extensions are represented by sets of *literals*: atomic formulas or negated atomic formulas. In order to test if a rule applies to a certain extension, the antecedent of the rule is broken down into its literal parts using De Morgan rules, and for these parts it is then checked if they are satisfiable by the extension. When a rule is applied, the consequent of the rule is also broken down into its literal subparts, and these are added to the extension.

Currently, there are two versions of the implementation. The first implements a single-extension BOID. It is a simple prioritized production system, which iterates through the following loop. First, find all rules that are applicable to the current extension: the *conflict set*. Second, select the rule with a minimal ρ value from the conflict set. Third, apply the selected rule to the extension, resulting in a new extension. This loop is repeated until either no more applicable rules are found, or until a fixed point has been reached. Because ρ assigns a unique priority to each rule, a single extension results.

The second implementation approximates a multi-extension BOID. This version does not make explicit use of the ρ priority value for each rule, but uses a four-letter ranking on components as in section 2.3. For example, in a 'BIDO' type agent, the beliefs component is applied, followed by one intention rule, feeding back the result to the B component again. Then D rules can be applied, each time feeding back the results to the I and B components as well. Finally, the O rules are applied, each time feeding back the results to the B, I and D components as well. Conflicts among rules within one component are solved by the top-to-bottom order of application used in Prolog. It is possible to back-track over these choices, producing different possible extensions. In a similar way, also partial agent types can be dealt with, producing even more alternative possible extensions.

There are alternative ways of implementing the prototype. Since the single-extension BOID makes use of a rather standard prioritization mechanism to deal with conflict resolution, many production systems woud be suitable to implement a BOID. An example is the CLIPS expert system shell, which uses the fast RETE algorithm for matching applicable rules [13]. For such imlementations, the BOID becomes a design heuristic, which helps the knowledge engineer to cluster rules into components and select a prioritization mechanism based on the desired agent type.

4.2 Examples

In this section, we illustrate how conflicts between attitudes can be solved within the BOID architecture and its corresponding control loop. To this end, we work out two examples presented in Section 2. Consider the example of a binary BD conflict introduced by Thomason [21]. This example can be represented by the following rules:

```
b_rule(ex1, true —> ~time_off).
b_rule(ex1, ~time_off —> ~vacation).
d_rule(ex1, true —> vacation).
```

Let the observations of a realistic agent be empty. We first derive all beliefs resulting in the following extension:

[~time_off, ~vacation]

This extension is input to the desire component. Because the only D-rule is not applicable, the final result remains the same. A non-realistic agent on the other hand would produce [vacation, ~time_off] as the final result. Such an agent clearly suffers from 'wishful thinking'.

Now, consider the more complex example of a quadruplicate conflict as given in Section 2. This example can be represented as follows.

```
b_rule(ex2, true —> expensive_ticket).
b_rule(ex2, ~too_much_money —> cheap_hotel &
                               ~expensive_ticket).
b_rule(ex2, ~too_much_money —> ~cheap_hotel &
                               expensive_ticket).
i_rule(ex2, true —> conference).
o_rule(ex2, conference —> ~too_much_money).
d_rule(ex2, true —> ~cheap_hotel).
```

Lets examine a social simple minded agent, of type 'BIOD'. Let the input of the agent be empty. Then, following the control loop, we first derive all beliefs and intentions, resulting in the following extension:

[conference, expensive_ticket]

Because it is a social agent, the obligation rule is applied first. This results in the following intermediate extension:

[~too_much_money, conference, expensive_ticket]

This extension is fed back into the B component where it triggers the third rule, because the second rules is not applicable as we already have expensive_ticket. This produces the following final extension:

[cheap_hotel, ~too_much_money, conference, expensive_ticket]

However, in a selfish agent of type 'BIDO', the D-rule would be applied first, resulting in the following final extension:

[~cheap_hotel, conference, expensive_ticket]

Note that sending the results back to the belief component does not make any difference here.

5. RELATED RESEARCH

Previous theoretical research has often neglected to show a possible implementation. Like [11, 6] we not only provide a theoretical framework, but we also provide an architecture which includes a control procedure in the style of Rao and Georgeff's BDI interpreter [17]. We extend BDI with obligations and conflict resolution.

Thomason [21] proposes a so-called BDP-logic for beliefs, desires and planning which is capable of modeling a wide range of common-sense practical arguments, and which can serve as a more general and flexible model for agent architectures. In Thomason's approach it is explicitly defined when desires override beliefs, whereas in our approach this is determined by the control loop. A detailed comparison can be found in [5].

Dignum *et.al.* [11] propose an alternative extension of BDI with obligations. It is based on on extension of the BDI interpreter with potential actions to reason about these effects.

6. CONCLUDING REMARKS

We have discussed possible conflict types that may arise within or among informational and motivational attitudes and we explained how these conflicts can be resolved within the BOID architecture. The resolution of conflicts is based on Thomason's idea of prioritization, which is implemented in the BOID architecture as the order of derivations from different types of attitudes. We have shown that the order of derivations determines the type of an agent. For example, deriving desire before beliefs produces wishful thinking agents and deriving obligations before desires produces super-social agents. In general, the order of derivation can be used to identify different types of agents.

An important ingredient in the BOID architecture is the presence of feedback loops. Through these feedback loops already derived beliefs, obligations, desires and intentions are sent back (at several stages) as new input to the BOID. These feedbacked inputs may trigger new beliefs, obligations, desires and even intentions. For example, an obligation to go to the assistance of your neighbors may induce the obligation to tell them you will come, and a desire to go to the dentist may induce the belief that pain will result – but of course not the desire that pain results.

Issues for further research are the methods for extension selection and its relation with planning and scheduling. It is possible that an intention is not immediately realized and that the future deliberations of the BOID can be influenced by these scheduled intentions. In the presented version of the BOID, all prior intentions are sent back via the feedback loop as well. These intentions can be overridden by other motivational attitudes such that the early derived intentions may need to be removed from the scheduled plans (intention reconsideration). Note also that adding a scheduling and planning component in the BOID architecture may give rise to the so-called delayed stimulus response behavior, in the sense that the BOID may be responding to an earlier observed stimulus. For example, agent A intends to go on vacation. He receives the information that his mother in law is hospitalized and therefore he has to visit her. After visiting her, he can return to the old intention and go on vacation.

In our opinion the gap between our proposed architectures and their underlying logics is much smaller than the very large gap between modal BDI logics and BDI architectures. We believe that the presented architecture and benchmark examples already provides some material to close the gap between theory and practice of agent design.

7. REFERENCES

[1] C. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

[2] J. Bell and Z. Huang. Dynamic goal hierarchies. In A. R. L. Cavedon and W. Wobcke, editors, *Intelligent Agent Systems, Theoretical and Practical Issues*, LNAI 1209, pages 88–103. Springer, 1997.

[3] C. Boutilier. Toward a logic for qualitative decision theory. In *Proceedings of the KR'94*, pages 75–86, 1994.

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

[5] J. Broersen, M. Dastani, and L. van der Torre. Wishful thinking. In *Proceedings of DGNMR'01*, 2001.

[6] C. Castelfranci, F. Dignum, C. Jonker, and J. Treur. Deliberative normative agents: principles and architecture. In *Proceedings of the ATAL'99*, LNCS, 1999.

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

[8] M. Dastani, Z. Huang, and L. van der Torre. Dynamic desires. In *Proceedings of ICMAS2000 Workshop on game-theoretic and decision-theoretic approaches to agency (GTDT'00)*, 2000.

[9] M. Dastani, J. Hulstijn, and L. van der Torre. BDI and QDT: a comparison based on classical decision theory. In *Proceedings of AAAI Spring Symposium GTDT'01*, 2000.

[10] F. Dignum. Autonomous agents and norms. *Artificial Intelligence and Law*, 7:69–79, 1999.

[11] F. Dignum, D. Morley, E. Sonenberg, and L. Cavedon. Towards socially sophisticated BDI agents. In *Proceedings of the ICMAS 2000*, pages 111–118, 2000.

[12] T. Eiter, V. Subrahmanian, and G. Pick. Heterogeneous active agents I: Semantics. *Artificial Intelligence*, 108 (1-2):179–255, 1999.

[13] C. L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Inteligence*, 19:17–37, 1982.

[14] Jones and Sergot. On the characterisation of law and computer systems: The normative systems perspective. In *Deontic Logic in Computer Science: Normative System Specification*, 1993.

[15] J. Pearl. From conditional oughts to qualitative decision theory. In *Proceedings of the UAI'93*, pages 12–20, 1993.

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

[17] A. Rao and M. Georgeff. An abstract architecture for rational agents. In *Proceedings of the KR92*, 1992.

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

[19] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[20] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[21] R. Thomason. Desires and defaults. In *Proceedings of the KR'2000*. Morgan Kaufmann, 2000.

[22] van der Torre and Tan. Contrary-to-duty reasoning with preference-based dyadic obligations. *Annals of Mathematics and Artificial Intelligence*, 27:49–78, 1999.

[23] L. van der Torre and Y. Tan. Diagnosis and decision making in normative reasoning. *Artificial Intelligence and Law*, 7:51–67, 1999.

[24] G. von Wright. *Practical Reason*. Basil Blackwell, Oxford, 1983.