An Alternative Classification of Agent Types based on BOID Conflict Resolution

Jan Broersen [a] Mehdi Dastani [b] Zhisheng Huang [a]
Joris Hulstijn [a] Leendert van der Torre [a]

[a] Vrije Universiteit Amsterdam
[b] University Utrecht

Abstract

In this paper, we introduce an alternative classification of agent types based on the BOID architecture. According to BOID, agents consist of at least four components called beliefs, obligations, intentions, and desires. The output of the different components may conflict and these conflicts are solved by an ordering function that determines in which order components should generate outputs. It is this ordering function that determines the type of agents.

1 Introduction

The type of an agent determines its behavior by specifying how it behaves in conflict situations, i.e. how the agent resolves conflicts between its mental attitudes. These agent types can be translated into properties of their architectures which are defined in terms of knowledge representation (i.e. structure) and reasoning mechanism (i.e. control). Various architectures for computational agents are proposed each of which results in a certain agent classification [4, 5]. In these agent architectures, it is assumed that the agent behavior is governed by the rational balance between its various mental attitudes.

In this paper, we introduce a classification of agent types based on how conflicts are resolved in the BOID architecture. To do this, we first explain some properties of the BOID architecture that are relevant for this classification. The architecture and the logic of BOID are discussed in more detail elsewhere [2, 3]. The BOID architecture contains at least four components representing agent's beliefs, obligations, intentions and desires. As these components output beliefs, obligations, intentions and desires only for certain inputs, they represent *conditional* mental attitudes. We discuss possible conflicts between the outputs of various mental attitudes and argue that these conflicts can be resolved by ordering the generations of the outputs. In BOID, conflict resolution mechanisms are understood as a generalization of the already existing concepts such as realistic, selfish, social and stable agents. Based on how conflicts are solved, we will introduce a classification of agent types.

The layout of the paper is as follows. In Section 2 the BOID architecture and its corresponding conflict resolution mechanism are introduced. In Section 3, possible agent types within the BOID architecture are discussed and it is explained how these agent types can be mapped to agent architectures. Then, in Section 4 an example of conflicts is formalized and it is shown how different agent types resolve these conflicts in different ways. In Section 5, we briefly discuss the BDI architecture [5] and compare this with our BOID architecture. Finally, we conclude the paper and indicate future research directions.

2 Agent Architecture

The agent type classification, which we will introduce in the next section, will be defined in terms of properties of the BOID architecture. Therefore, we first briefly explain this architecture, which can be seen as a black box with observations as input and intended actions as output. The agent observes the environment and reacts to it by means of detectors and effectors, respectively. In the BOID architecture agent's mental attitudes such as beliefs, obligations, intentions, and desires are mapped to components within the agent architecture, in the sense that each component outputs one of the mental attitudes. The behavior of each component is specified by propositional logical formulas, 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 [6] these are called *extensions*. In this paper, we consider a BOID architecture that generates only one extension. The logic that specifies the extension is parameterized with an ordering function ρ that resolves conflicts. This function constraints the order of derivation steps for different components and characterizes the type of the agent. We first briefly discuss the BOID logic and then explain how the ordering function can be used to defined various agent types.

2.1 Logic or Calculation Scheme

The inputs and outputs of each component in the BOID architecture are extensions, i.e. sets of logical formulas closed under logical consequence. In this section, we give a constructive definition of extensions by specifying how they can be calculated. The calculation starts with a set of observations W. Unlike normal beliefs, which may have a default character, observations cannot be overridden. We assume initial sets of defeasible rules for the other components: B,O, I, D. We also assume an ordering function ρ that is defined on the rules from different components. In case of multiple applicable rules, the one with the lowest ρ value is applied. 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

$$
\begin{aligned}
E_0 &= W \qquad \textit{and for } i \geq 0 \\
E_{i+1} &= Th_L(E_i \cup \{w \mid (\alpha \hookrightarrow w) \in B \cup O \cup I \cup D \textit{ and} \\
&\quad (\alpha \hookrightarrow w) \textit{ is applicable to } E_i \textit{ and} \\
&\quad \nexists(\beta \hookrightarrow v) \in B \cup O \cup I \cup D \textit{ applicable to } E_i \\
&\quad \textit{such that } \rho(\beta \hookrightarrow v) < \rho(\alpha \hookrightarrow w) \ \}\ ).
\end{aligned}
$$

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

We assume that ρ assigns values to the rules of all components such that all rules from one component have either smaller or greater values than the rules from another component. It is important to note that this assumptions induces an ordering among components. We believe that it does not make any sense to consider those ρ's that do not induce any ordering among components, as it will be explained later in this paper. This assumption is formally defined as follows:

Definition 2 *Let B, O, I, and D be the mutually exclusive sets of rules for beliefs, obligations, 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 in practice, full extensions are not 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 Agent Types

An agent type is considered here as 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 other component; otherwise, as we will explain later, unrealistic agent types result. This reduces the number of possible overruling orders to 6. Some examples of conflict resolution with beliefs are as follows. A conflict between a belief and an intention means that an intended action can no longer be executed due to the changing environment. Beliefs therefore overrule the intention, which is retracted. Any derived consequences of this intention are retracted too. Of course, one may allow intentions to overrule beliefs, but this results in unrealistic behavior. Also, as observed by Thomason [6], the belief must override desires or otherwise there is wishful thinking; the same argument applies to obligations and intentions as well. Moreover, a conflict between an intention and an obligation or desire means that you now should or want to do something else than you intended before. Here intentions override the latter because it is exactly this property for which intentions have been introduced: to bring stability [1]. 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 and thus as representing the agent type, a realistic agent can have any of the following six specific agent types, i.e. BOID, BODI, BDIO, BDOI, BIOD, and BIDO. Note that we overloaded the name BOID in this way, because it becomes a specific type of agent (i.e. realistic social stable agent) as well as the general name for the agent architecture. These six specific agent types, in which beliefs override all other components, can be represented as a constraint on the ρ function resulting in a general agent type, called realistic. This constraint can be formulated as:
$\rho(r_b) < \rho(r_o) \ \wedge \ \rho(r_b) < \rho(r_i) \ \wedge \ \rho(r_b) < \rho(r_d)$
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.

Now that we have a specific ρ function that characterizes realistic BOID type, we will indicate how the extension is calculated. Following definition 1, a realistic BOID agent starts with the observations and calculate a belief extension by iteratively applying belief rules. When no belief rule is applicable anymore, then 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 another component is chosen again. 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.

Other general agent types can be specified as constraints on the ρ function as well. Since we consider in this paper only realistic agent types, we will limit ourselves to those general agent types that are subtypes of realistic agent types. Some of these general agent types can now be represented as follows.

BIDO and BIOD are called (realistic) *stable*: after beliefs, intentions have the highest priority and overrule everything else, i.e.
$\mathit{Realistic} \ \wedge \ \rho(r_i) < \rho(r_d) \ \wedge \ \rho(r_i) < \rho(r_o)$.

BDIO and BDOI are called (realistic) *selfish*: after beliefs, desires have the highest priority and overrule everything else, i.e.
$\mathit{Realistic} \ \wedge \ \rho(r_d) < \rho(r_o) \ \wedge \ \rho(r_d) < \rho(r_i)$.

BOID and BODI are called (realistic) *social*: after beliefs, obligations have the highest priority and overrule everything else, i.e.
$\mathit{Realistic} \ \wedge \ \rho(r_o) < \rho(r_i) \ \wedge \ \rho(r_o) < \rho(r_d)$.

Other agent types are still possible. A hierarchy of these and other agent types is illustrated in Figure 1. The level in this hierarchy indicates the generality of agent types. In this type hierarchy, we use terms such as (realistic) *social stable* and (realistic) *stable social* as referring to specific and different agent types: the first prefers O above I and I above D while the second prefers I above O and O

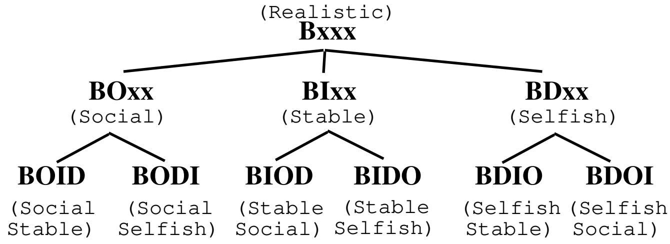

Figure 1: The hierarchy of agent types. .

above D. Although these names may be confusing, they are used to reflect the underlying ordering idea. Other names that correspond or reflect these orderings can be used instead. Note also that in this hierarchy we write *Selfish* instead of *Realistic Selfish* and *Social Selfish* instead of *Realistic Social Selfish*.

3.1 Mapping Agent Types to Agent 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 can be mapped to a certain agent architecture. For example, consider the realistic social stable agent type (i.e. BOID) that is specified by a total derivation ordering. This agent type is mapped to the architecture illustrated in Figure 2-A. This architecture 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 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. The architecture illustrated in Figure 2-A represents a total ordering among mental attitudes since the connections between the components fully determine how extensions flow around. However,

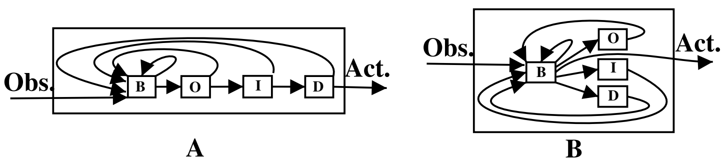

Figure 2: A) Realistic social stable, and B) Realistic agent architecture.

agent types that are specified by partial derivation orderings can also be mapped to agent architectures. As a consequence of partiality of the derivation order, at least one component in such architecture is connected to two other components. For example, consider the realistic agent type which implies that the belief component overrules all other components. The order in which the rules from O, I, and D components are applied are not determined. The resulting general agent architecture that represents realistic agent type is illustrated in Figure 2-B. 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 arbitrary to O, I, or D component. Note that other agent types that are represented by a ρ function can be mapped to BOID architecture in the same way.

4 Examples

In this section, we illustrate how conflicts between mental attitudes can be solved within the BOID architecture by giving an example that describes the following mental attitudes: *If I go to Montreal, then I believe that there are no cheap rooms close to the conference site. If I go to Montreal, then I am obliged to take a cheap room. If I go to Montreal, then I desire to stay close to the conference site. I intend to go to Montreal.* This example can be represented by the following rules:

1) b_rule(montreal ∧ cheapRoom—> ∼closeConfSite).
2) b_rule(montreal ∧ closeConfSite—> ∼cheapRoom).
3) d_rule(montreal—> closeConfSite).
4) o_rule(montreal —> cheapRoom).
5) i_rule(true —> montreal).

Lets examine a simple minded social agent, i.e. BIOD. Let the input of the agent be empty. Then, following the extension calculation mechanism, we first derive all beliefs and intentions, resulting in the following extension:

[montreal]

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

[montreal, cheapRoom]

This extension is fed back into the B component where it triggers the first belief rule (i.e. the first rule), because the second belief rules is not applicable as we already have cheapRoom. This produces the following final extension:

[montreal, cheapRoom, ∼closeConfSite]

This extension denotes the situation in which agent have decided to go to Montreal and takes a cheap room not close to the conference site; indeed a social behavior. However, in a selfish agent of type 'BIDO', the D-rule would be applied before any obligation rule is applied, resulting in the following final extension:

[montreal, closeConfSite,∼cheapRoom]

Note that sending the results back to the belief component does not make any difference here. This extension denotes the situation in which agent have decided to go to Montreal and takes an expensive room close to the conference site; indeed a selfish behavior.

5 Related Research

The BDI architecture [5] is probably the most appreciated agent architecture. It is based on possible-world formalism where beliefs, desires and intentions are considered as independent mental attitudes formalized by different modal operators. These attitudes and their relations are characterized by constraints formalized as axioms. For example, the belief is characterized by $KD45$ axioms where the D rule guarantees that beliefs cannot be inconsistent. Furthermore, the specific relationships between these modalities are given by two types of constraints called static and dynamic constraints. For example, the static stability constraint, formalized by the axiom $Int(p) \rightarrow Bel(p)$, states that if it is intended to bring about p, then p should be believed. The dynamic constraints capture the dynamics of these relationships over time. For example, the single-mindedness axiom, formalized as $A(Int_a(A\Diamond\alpha)U(Bel_a(\alpha) \vee \neg Bel_a(E\Diamond\alpha)))$, states that agent a should give up to maintain the goal α as soon as the agent believes that α is not attainable anymore.

The number of mental conflicts in BOID is larger than in BDI since BOID extends BDI with obligations. Unlike BDI, BOID resolves conflicts by means of ordering of derivations. BOID has the advantage of bridging the large gap between agent logic and its corresponding architecture, as shown by the simple mappings between agent types and architectures in section 3. However, we believe that the BOID derivation ordering is much less expressive than the BDI formalism such that only few constraints, which can be formalized by the BDI axioms, can be formalized in BOID. For example, the static stability constraint can be realized by an ordering function that overrides intentions by beliefs. In contrast, dynamic constraints can hardly be realized by the derivation ordering. Perhaps, the only dynamic constraint that can be modelled by the derivation ordering is when (prior) intentions override obligations and desires. The captured dynamics is related to the time-related nature of (prior) intentions.

Note that although BOID does not account for time, it is supposed to be situated in a dynamic environment. It receives input from the environment, calculate the 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 general control loop for the BOID architecture, which determines the behavior of the agent in dynamic environment. This control loop can be written as follows:

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

6 Future Research and Concluding Remarks

We have discussed possible conflict types that may arise within or among mental 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 mental attitudes. We have shown that the order of derivations determines the type of an agent. For example, deriving desire before beliefs produces wishful thinking (i.e. unrealistic) agents and deriving obligations before desires produces social agents. In general, the order of derivation can be used to identify different types of agents.

We admit that planning is an essential part of any agent architectures which is not considered in this paper. In fact, we believe that planning component influence the computation of extensions and therefore may play an important role in agent type classification. For example, when a generated extension cannot be transformed into a sequence of actions, another extension should be calculated. The exact choice for a new extension should depends on the type of agent. We also believe that the way the BOID components are updated during a control loop may depend on the type of agent as well. The integration of planning into BOID and the mechanism of updating various components have the highest priority in our research agenda.

References

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

[2] J. Broersen, M. Dastani and L. van der Torre. Resolving conflicts between beliefs, obligations, intentions and desires. *Symbolic and Quantitative Approaches to Reasoning and Uncertainty.* Proceedings of the ECSQARU'01, Lecture Notes in Computer Science, Springer Verlag, 2001.

[3] J. Broersen, M. Dastani, Z. Huang, J. Hulstijn and L. van der Torre. The BOID Architecture: Conflicts between beliefs, obligations, intentions, and desires. Proceedings of Fifth International Conference on Autonomous Agents (AA'01), "9–16",ACM Press (2001)

[4] J. P. Müller. *A conceptual model of agent interaction.* In Deen, S. M., editor, Draft proceedings of the Second International Working Conference on Cooperating Knowledge Based Systems (CKBS-94), pages 389-404, DAKE Centre, University of Keele, UK, 1994.

[5] 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.

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