Beliefs, Obligations, Intentions, and Desires as Components in an Agent Architecture

Jan Broersen,[1,*] Mehdi Dastani,[1,†] Leendert van der Torre[2,‡]
[1]Department of Information and Computing Sciences, Universiteit Utrecht, P.O. Box 80.089, NL-3508 TB Utrecht, The Netherlands
[2]Centrum voor Wiskunde en Informatica, P.O. Box 94079, NL-1090 GB Amsterdam, The Netherlands and Delft University of Technology, P.O. Box 5031, NL-2600 GA Delft, The Netherlands

In this article we discuss how cognitive attitudes like beliefs, obligations, intentions, and desires can be represented as components with input/output functionality. We study how to break down an agent specification into a specification of individual components and a specification of their coordination. A typical property discussed at the individual component specification level is whether the input is included in the output, and a typical property discussed at the coordination level is whether beliefs override desires to ensure realism. At the individual level we show how proof rules of so-called input/output logics correspond to properties of functionality descriptions, and at the coordination level we show how global constraints coordinating the components formalize coherence properties. © 2005 Wiley Periodicals, Inc.

1. INTRODUCTION

The structured approach to the specification of agent systems based on a notion of compositional architecture has been called *compositional design*, and an example of a compositional design method is the DEsign and Specification of Interacting REasoning components (DESIRE) developed by Treur and colleagues.[1] The reasoning process is structured according to a number of reasoning components that interact with each other. Components may or may not be composed of other components, where components that cannot be decomposed are called primitive components. The functioning of the overall agent system is based on the functionality of these primitive components plus the composition relation that coordinates their interaction. Specification of a composition relation may involve, for example, the possibilities for information exchange among components and the control that activates the components. The functionality of a (primitive) reasoning

*Author to whom all correspondence should be addressed: e-mail: broersen@cs.uu.nl.
†e-mail: mehdi@cs.uu.nl.
‡e-mail: torre@cwi.nl.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 20, 893–919 (2005)
 • DOI 10.1002/int.20098

component within a compositional architecture can be described by Treur's functionality descriptions.[2] Typical properties are whether the input of a component is included in its output, or whether it supports reasoning by cases.

Our research question is how to represent cognitive attitudes by components such that an agent specification is broken down into a specification of individual components and the specification of their coordination (following a divide and conquer strategy). In this article we relate properties of Treur's functionality descriptions to the validity of proof rules in Makinson and van der Torre's input/output logics.[3,4] These logics have been developed as a general theory of propositional input/output operations as processes resembling inference, but where input propositions are not in general included among outputs, and the operation is not in any way reversible. We show that an input/output logic, characterized by a set of proof rules, specifies the input/output behavior of reasoning components as described by the functionality descriptions of those components. Functionality descriptions can be specified in input/output logic, thus enabling a *logical* specification of reasoning components. The correspondence between Treur's functionality descriptions and the input/output logics relates functionality descriptions to cognitive agent architectures, because, according to Makinson and van der Torre, examples of the input/output operations they study "arise in contexts of conditional obligations, goals, ideals, preferences, actions, and beliefs."[3]

We also study the coordination of components that represent cognitive attitudes. The coherence among cognitive components can be described by various kinds of coordination principles. For example, it may make sense for belief sets to include the observations, but it makes less sense for obligation or desire sets to include the beliefs. You may desire to go to the dentist and believe that going to the dentist implies pain, but this does not imply the desire for pain.[5] Other coordination principles are encoded in the overall conflict resolution strategy in the architecture's control specification. A typical property is realism, which has been formalized as the overriding of desires by beliefs.[6,7]

Our motivation to study how cognitive attitudes can be allocated in an agent architecture is to give formal foundations for several agent architectures such as Castelfranchi et al.'s deliberative normative agent architecture[8] and Broersen et al.'s BOID architecture.[9] The BOID architecture extends Thomason's BDP logic,[6] based on a planning component and a conflict resolution component for conditional beliefs and desires, with conditional obligations and prior intentions, borrowed from, respectively, deontic action programs[10] and BDI systems.[11,12]

The layout of this article is as follows. In Section 2 we discuss the design of component-based agents, with the BOID agent as an example. In Section 3 we consider the specification of individual components, and in Section 4 we consider the coordination of the components.

2. DESIGNING COMPONENT-BASED AGENTS

In the study of architectures for autonomous agents, cognitive attitudes have been represented as components. A typical example of this allocation is the BOID

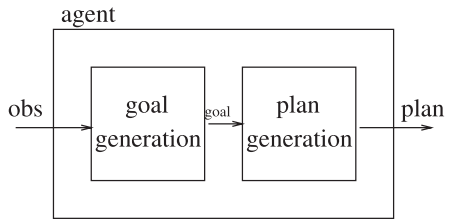

Figure 1. Goal-based agent.

architecture,[9] which contains components outputting beliefs (B), obligations (O), intentions (I), and desires (D). In this section we discuss how to design a component-based BOID agent. Design problems are iteratively replaced by a number of smaller design problems together with the problem of how the solutions are composed based on coordination principles, which leads to a transparent structure of the design and to support for reuse and maintainability of components and generic models.

Figure 1 visualizes the general goal-based agent architecture studied in Ref. 13 that consists of a goal generation and a plan generation component. The agent can be seen as a black box with observations (obs) as input and intended plans or actions as output. As usual, the agent is assumed to sense the environment by detectors and execute actions by effectors. The design problem of the agent is now reformulated as designing the goal generation component, the plan generation component, and their coordination. This design suggests sequential composition, which may or may not be possible for a specific problem or agent. We therefore do not restrict ourselves to sequential composition and also allow for other compositional relations.

Moreover, Figure 2 visualizes the goal generation component studied in Refs. 7 and 9, which contains the four components *beliefs* (B), *obligations* (O), *intentions* (I), and *desires* (D). The design problem of the goal generation module decomposes into the problem of how to model the input/output behavior of the agent's cognitive attitudes and the problem of their interaction. Again we do not restrict ourselves to sequential composition. This is visualized by feedback loops, which have subtle implications.

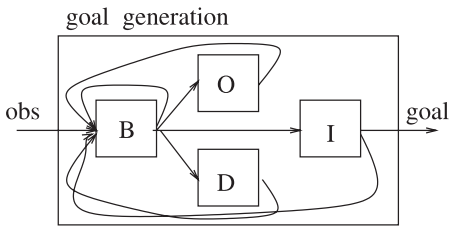

Figure 2. Goal generation component of goal-based agent.

Finally, these four cognitive components are primitive components, because they are not further broken down. We could have further distinguished among distinct types of beliefs, distinct types of obligations, and so on. The components can be implemented in a variety of ways. For example, the belief component may maximize cross entropy or apply AGM belief revision,[14] and its output may be a probability distribution, a set of them, plausibility measures, a belief set, and so forth. In a logic-based goal generation component like the BOID architecture, the input (the observations) and the output (the goals) are sets of logical formulas. The behavior of each component as well as the interaction among the components are described as follows.

Component specification. Each component can be described by (or may even consist of) a set of rules—pairs of logical formulas—and the input and the output of each component are sets of formulas. For example, the belief component may be described by the set of rules $\{\top \hookrightarrow rain, rain \hookrightarrow wet\}$, representing the beliefs that "it rains" and "if it rains, then streets are wet." The belief component outputs $\{rain\}$ without any input (empty observation, $\emptyset$) and it outputs $\{rain, wet\}$ if it receives $\{rain\}$ as input. In the BOID architecture, the output of a BOID component is its input together with the body of applicable rules.

Component coordination specification. At any point in time, the feedback connections guarantee that the components can receive outputs, which are generated so far, in order to generate new outputs. For example, in Figure 2, there is a feedback connection from the output interface of the belief component to its input interface, guaranteeing that the output of the belief component is used to generate new belief outputs. Similarly, the output interfaces of O, I, and D components are connected to the input interface of the B component by feedback connections. The outputs of O, I, and D components are sent back to the belief component through these feedback connections.

A tool for the design of individual components is Treur's so-called functionality description,[2] described in more detail in Section 3, which, according to Treur, is independent of the component's specific internal knowledge representation, inference relations, or implementation. The component's internal structure can be changed as long as its functionality remains the same. In this way a functionality description obtains not only a clear and well-defined analysis of the component's behavior, but it also supports reusability and maintainability, as well as information hiding.

A typical example of a phenomenon that can be defined on the individual component level as well as on the composition level is the resolution of conflicts. It is well known from the work on BDI agents that the type of an agent determines its behavior; we can thus refer to it as an abstract behavior type.[15] The agent type is governed by the specific way it handles the rational balance among its cognitive attitudes such as beliefs, obligations, intentions, and desires. For example, the type of an agent determines how it resolves conflicts within and among its cognitive attitudes. In the BOID architecture the rational balance such as resolving conflict

among cognitive attitudes is determined based on the properties of their architectures, which are in turn defined in terms of data representation and a reasoning mechanism. Two types of conflicts can be distinguished in the goal generation component in Figure 2.

Component specification: Conflicts within a component. Conflicts and their resolutions are analyzed within the component itself and are therefore not visible from the outside of the components. The conflict resolution behavior within a component can be specified in terms of properties of functionality descriptions, that is, properties of input/output behavior of the component.

Component coordination specification: Conflicts among components. Conflicts can be formalized as conflicts among outcomes of functionality descriptions. These conflicts are addressed in the possibly dynamic composition relation. There are six total orders, the number in which the three letters O/I/D can be ordered, which expresses that

- The agents are realistic and the belief therefore overrides all other mental attitudes.
- Intentions may override desires and obligations (stable agents) or the other way around (unstable agents).
- Desires may override obligations (selfish) or the other way around (social).

Besides the six total orders there are also many partial orders, in which, for example, there is no general preference of desires over obligations or vice versa.

For further information on the BOID project see the BOID's home page.[16]

3. INDIVIDUAL COMPONENT SPECIFICATION

In Section 3.1 we discuss the use of functionality descriptions in compositional design to formalize the functionality of a primitive component within a compositional architecture, and in Section 3.2 we relate this notion to proof rules of input/output logics.

3.1. Functionality Descriptions . . .

As part of the foundations of compositional architectures, the notion of functionality description has been introduced in Ref. 2. A functionality description is the set of input/output combinations provided by the component, that

- gives a *semantical description* of the reasoning that provides conclusions "for the outside"—for other components outside the reasoning component—as output, but at the same time uses information "from the outside" as input. In this sense we can say that such functionality describes the *interactive* role of a given component.
- may be seen as a *declarative description*, in the sense that a functionality description abstracts from the dynamic aspects of the component's reasoning process. It only describes which facts can be derived in a given situation, and not at what time and in which order specific facts are derived.

The remainder of this section repeats some definitions of Ref. 2. The signature is based on a distinction between input, intermediate, and output signatures, which are assumed to be finite.

DEFINITION 1 (Ref. 2, Def. 3.1). *A propositional signature Σ is a three-tuple, $\langle InSig(\Sigma), IntSig(\Sigma), OutSig(\Sigma)\rangle$, where $InSig(\Sigma)$, $IntSig(\Sigma)$, and $OutSig(\Sigma)$ are ordered sets of atom names, respectively called the input signature, the internal signature, and the output signature. The input and output signatures can contain common names, but the internal signature is disjoint from them. All signatures are assumed to be finite.*

The semantics is based on partial models. Reasoning components are able to draw partial conclusions if a partial input information state is given. Therefore the formal description of the declarative functionality of a component also treats partiality of information both at the input side and the output side of the component.

DEFINITION 2 (Ref. 2). *A partial model M of signature Σ is an assignment of truth values from $\{1,0,u\}$ to the atoms of Σ. By $M(a)$ we denote the truth value assigned to atom a. We call M a complete model if for all atoms a the truth value $M(a)$ is not u. Moreover, we call M a complete input model if for all atoms a of $InSig(\Sigma)$ the truth value of a is not u, but for all atoms a of $\Sigma \backslash InSig(\Sigma)$, the truth value of a is u. The model $Out(M)$ is obtained from M such that for all atoms a of $OutSig(\Sigma)$ the truth value of a is its truth value in M, but for all atoms a of $\Sigma \backslash OutSig(\Sigma)$, the truth value of a is u.*[a]

An important notion is that a model can refine another model of the same signature, which means that if the less refined model assigns true or false to propositions, the more refined model assigns the same truth value to those propositions, but when the less refined model assigns unknown to propositions, the more refined model may assign true or false. This notion is used to define the greatest common information state, which may be seen as the maximal information on which all members of the set of information states agree. The greatest common information state of the set of information states $gci(V)$ can be constructed as follows: for any atom a on which all members of V agree, take this truth value, and if the members of V disagree, take the truth value u.

DEFINITION 3 (Ref. 2). *The refinement relation $\leq$ between partial models of the same signature is defined by $M \leq N$, if for every atom a it holds that $M(a) \leq N(a)$ (i.e., point by point), where the partial order of truth values is defined by $u < 1$ and $u < 0$. Let V be a nonempty set of partial models of signature Σ. By $P(V)$ we denote the set of all partial models that can be refined to a model in V. The greatest*

[a]Treur's definitions of complete input model and the model $Out(M)$ are obtained via reductions and expansions, but because we are only interested in the notions we need to define functionality descriptions, we do not give the details.

common information state of V is the partial model $N \in P(V)$*, denoted by* $N = gci(V)$*, such that for all* $M \in V$*, it holds that* $N \leq M$*, and for any* N' *satisfying this condition, it holds that* $N' \leq N$*.*

We have now given the necessary machinery to define functionality descriptions. Treur also defines additional properties of functionality descriptions, such as regularity, which we do not discuss in this article.

DEFINITION 4 (Ref. 2, Def. 4.1). *Suppose a signature* Σ *is given, a nonempty set of complete input models* W_{in} *for* Σ*, and a mapping* $\alpha : P_{in} \rightarrow P$*, where* $P_{in} = P(W_{in})$ *and P is the set of partial models for* Σ.[b]

(1) The mapping α *is called* conservative *if for all* $M \in P_{in}$ *it holds that*

$$M \leq \alpha(M)$$

(2) The mapping α *is called* monotonic *if for all* $M, N \in P_{in}$*, it holds that*

$$M \leq N \Rightarrow \alpha(M) \leq \alpha(N)$$

(3) The mapping α *is called* self-bounded *if for all* $M, N \in P_{in}$ *it holds that*

$$M \leq \alpha(N) \Rightarrow \alpha(M) \leq \alpha(N)$$

(4) The mapping α *is called* well-informed *if for all* $M \in P_{in}$ *it holds that*

$$out(\alpha(M)) = gci(\{out(\alpha(N)) \mid N \in W_{in} \& M \leq N\})$$

A partial model can be represented by a finite set of propositional literals and therefore by a propositional formula. In this representation, the refinement relation represents propositional derivability. For example, a complete model that assigns true (1) to a and false (0) to b refines a partial model that assigns unknown (u) to a and false (0) to b, because $a \wedge \neg b$ logically implies $\neg b$.

Moreover, in this interpretation we can interpret α as a kind of logical consequence operator. However, it is a consequence operator of an unusual kind, as α does not satisfy the Tarskian properties. For example, α is not necessarily conservative and the associated consequence relation therefore does not necessarily satisfy identity. In the following section we therefore relate functionality descriptions to input/output logic, as this logic does not necessarily satisfy the identity rule either.

3.2. … and Input/Output Logics

Input/output logic (IOL)[3,4] is a theory of input/output operations resembling inference, but where input propositions are not, in general, included among outputs and the operation is not in any way reversible. Makinson and van der Torre

[b]Treur says "P is a set of partial models for Σ."

write *out* instead of *output*, but we write *output* to avoid confusion with Treur's notion of *out*. Moreover, Makinson and van der Torre define input and output of their operations as arbitrary formulas, but to represent the functionality descriptions, here we restrict ourselves to conjunctions of sets of literals. To facilitate the semantics of the output operations Makinson and van der Torre extend the generating set G with $(\top, \top)$ such that it is never empty. Because we do not consider semantics of input/output logics in this article we also do not consider this borderline case.

DEFINITION 5 (Input/Output Logic).[3] *Let L be the fragment of propositional logic that contains all conjunctions of literals, $\vdash$ derivability in L, G be a set of pairs of conjunctions of propositional literals $\{(a_1, x_1), \ldots, (a_n, x_n)\}$, read as "if input a_1 then output x_1," and so forth, and consider the following proof rules strengthening of the input (SI), weakening of the output (WO), conjunction for the output (AND), disjunction of the input (OR), cumulative transitivity (CT), transitivity (TRANS), and identity (Id) defined as follows:*

$$\frac{(a,x)}{(b,x)}\ b \vdash a\ (SI) \qquad \frac{(a,x)}{(a,y)}\ x \vdash y\ (WO) \qquad \frac{(a,x)(a,y)}{(a,x \wedge y)}\ (AND)$$

$$\frac{(a \wedge b,x)(a \wedge \neg b,x)}{(a,x)}\ (OR) \qquad \frac{(a,x)(a \wedge x,y)}{(a,y)}\ (CT) \qquad \frac{(a,x)(x,y)}{(a,y)}\ (TRANS)$$

$$\frac{}{(a,a)}\ (Id)$$

Each subset of the above rules together with a "silent" rule that replaces propositional formulas by logical equivalent ones, defines an output operation $output(G)$ as a closure operation on G. Makinson and van der Torre focus on the following eight output operators.

- *$output_1$: SI + AND + WO (simple-minded output)*
- *$output_2$: SI + AND + WO + OR (basic output)*
- *$output_3$: SI + AND + WO + CT (simple-minded reusable output)*
- *$output_4$: SI + AND + WO + OR + CT (basic reusable output)*
- *out_i^+: out_i + Id (throughput)*

Moreover, a mapping from propositions to propositions is defined indirectly by $x \in output(G,a)$ iff $(a,x) \in output(G)$.

To relate the functionality description with input/output logic, we assume that if $N = \alpha(M)$, then M is a description of the input, and N is a *complete* description of the output. We ignore the fact that functionality descriptions have

been restricted to a set of complete input models called domain descriptions W_{in}. The extension of the results below for a given domain description is straightforward.

DEFINITION 6. *Let Σ be a propositional signature as introduced in Definition 1, L the set of conjunctions of literals from Σ, and L_{in} the set of conjunctions from literals built from $InSig(\Delta)$. For a partial model M, let $\tau(M)$ be the following conjunction of literals:*

$$\tau(M) = \bigwedge \{a \,|\, M(a) = 1, a \in \Sigma\} \wedge \bigwedge \{\neg a \,|\, M(a) = 0, a \in \Sigma\}$$

The input/output relation associated with α is defined as follows: $(a, x) \in output_\alpha$ iff $\exists M \in P(W_{in}), N \in P$ such that $N = \alpha(M)$, $a \equiv \tau(M)$, and $\tau(N) \vdash x$. Moreover, a mapping from propositions to propositions is defined indirectly by $x \in output_\alpha(a) \Leftrightarrow (a, x) \in output_\alpha$.

The $output_\alpha$ operation is an input/output relation closed under the two rules AND and WO. It can be interpreted as a nonmonotonic weakening of Makinson and van der Torre's input/output logics, as compared to their systems the closure operator does not necessarily satisfy strengthening of the input.

THEOREM 1. *We have*

(1) $output_\alpha$ satisfies AND and WO.
(2) For any input/output operation satisfying AND and WO, there is a functionality description α such that this input/output operation is identical to $output_\alpha$.

Proof (sketch). Item 1 follows from Definition 6, where "$\tau(N) \vdash x$" defines output as logically closed set. Item 2 follows from lack of properties of α and can be proven by construction: Assume an arbitrary output relation and construct α. ■

Before we present our main theorem, we first relate the refinement relation on partial models to propositional derivability. For conservative, monotone, and well-founded, we only need the simple properties of Lemma 1.

LEMMA 1. *We have*

(1) $M \leq N$ iff $\tau(N) \vdash \tau(M)$
(2) $M \leq \alpha(N)$ iff $\tau(M) \in output_\alpha(\tau(N))$
(3) $\alpha(M) \leq \alpha(N)$ iff $output_\alpha(\tau(M)) \subseteq output_\alpha(\tau(N))$.

Proof. Follows directly from the definition of τ in Definition 6. ■

For the property of well-informed we need an inductive argument, because it is based on the greatest common information state of a set of models.

LEMMA 2. *We have*

(1) $\forall M \in P_{in} : Out(\alpha(M)) \leq gci(\{Out(\alpha(N)) | N \in W_{in} \wedge M \leq N\})$ *iff*
$\forall M \in P_{in} : Out(\alpha(M)) \leq gci(\{Out(\alpha(N)) | N \in P_{in} \wedge M \leq N\})$
(2) $\forall M \in P_{in} : Out(\alpha(M)) \leq gci(\{Out(\alpha(N)) | N \in P_{in} \wedge M \leq N\})$ *iff*
$\forall M, N \in P_{in} : Out(\alpha(M)) \leq Out(\alpha(N))$
(3) $\forall M \in P_{in} : gci(\{Out(\alpha(N)) | N \in P_{in} \wedge M \leq N\} \leq Out(\alpha(M)))$ *iff*
$\forall M \in P_{in} \forall p \in InSig(\Delta) : \tau(Out(\alpha(M)) \vdash output_{\alpha}(\tau(M) \wedge p) \vee output_{\alpha}(\tau(M) \wedge \neg p))$

Proof (sketch). Items 1 and 2 follow directly from the definitions. Item 3 follows from fact that $\leq$ is a semi-lattice, and *gci* is the infimum[d] of this semi-lattice. ■

We now relate the four properties of functionality descriptions to proof rules of input/output logic. There is one final complication. Functionality descriptions are only defined on partial models. Consequently, $output_{\alpha}$ is only defined for *consistent* formulas, that is, sets of literals that do not contain a propositional atom and its negation.

THEOREM 2. *Let the set of consistent propositional formulas contain the conjunctions of sets of literals where these sets do not contain an atom and its negation. For any functionality description* α*, the following hold:*

(1) α *is conservative iff* $output_{\alpha}$ *satisfies identity (Id) for consistent input formulas.*
(2) α *is monotonic iff* $output_{\alpha}$ *satisfies strengthening of the input (SI) for consistent input formulas.*
(3) α *is self-bounded iff* $output_{\alpha}$ *satisfies transitivity (TRANS).*
(4) α *is well-informed iff* $output_{\alpha}$ *satisfies the strengthening of the input (SI) and the disjunction rule (OR) for input formulas.*

Proof. Let L and L_{in} be set of propositional formulas as defined in Definition 6.

(1) α is conservative iff $output_{\alpha}$ satisfies identity (Id) for consistent input formulas:

$$\dfrac{\dfrac{\dfrac{\forall M \in P_{in} : M \leq \alpha(M)}{\forall M \in P_{in} : \tau(M) \in output_{\alpha}(\tau(M))}\ \text{Lem. 1}}{\forall a \in L_{in} : a \in output_{\alpha}(a)}\ \text{Def. 6}}{\forall a \in L_{in} : (a, a) \in output_{\alpha}}\ \text{Def. 6}$$

[d]The infimum is $inf(V) = \{M | \forall N \in V, M \leq N, \nexists M' > M, \forall N \in V, M' \leq N\}$.

(2) α is monotonic iff $output_\alpha$ satisfies strengthening of the input (SI) for consistent input formulas:

$$\dfrac{\dfrac{\dfrac{\dfrac{\forall M,N \in P_{in} : M \leq N \Rightarrow \alpha(M) \leq \alpha(N)}{\forall M,N \in P_{in} : \tau(N) \vdash \tau(M) \Rightarrow (output_\alpha(\tau(M)) \subseteq output_\alpha(\tau(N)))}\text{Lem. 1}}{\forall a,b \in L_{in} : b \vdash a \Rightarrow (output_\alpha(a) \subseteq output_\alpha(b))}\text{Def. 6}}{\forall a,b \in L_{in} : b \vdash a \Rightarrow (\forall x \in L : (a,x) \in output_\alpha \rightarrow (b,x) \in output_\alpha)}\text{Def. 6}$$

(3) α is self-bounded iff $output_\alpha$ satisfies transitivity ($TRANS$):

$$\dfrac{\dfrac{\dfrac{\dfrac{\forall M,N \in P_{in} : M \leq \alpha(N) \Rightarrow \alpha(M) \leq \alpha(N)}{\forall M,N \in P_{in} : \tau(M) \in output_\alpha(\tau(N)) \Rightarrow (output_\alpha(\tau(M)) \subseteq output_\alpha(\tau(N)))}\text{Lem. 1}}{\forall a,b \in L_{in} : b \in output_\alpha(a) \Rightarrow (output_\alpha(b) \subseteq output_\alpha(a))}\text{Def. 6}}{\forall a,b \in L_{in} : (a,b) \in output_\alpha \Rightarrow (\forall x \in L : (b,x) \in output_\alpha \rightarrow (a,x) \in output_\alpha)}\text{Def. 6}$$

(4) α is well-informed iff $output_\alpha$ satisfies the following two rules. First it satisfies the strengthening of the input (SI) for out formulas:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall M \in P_{in} : Out(\alpha(M)) \leq gci(\{Out(\alpha(N)) \mid N \in W_{in} \wedge M \leq N\})}{\forall M \in P_{in} : Out(\alpha(M)) \leq gci(\{Out(\alpha(N)) \mid N \in P_{in} \wedge M \leq N\})}\text{Lem. 1}}{\forall M,N \in P_{in} : M \leq N \Rightarrow Out(\alpha(M)) \leq Out(\alpha(N))}\text{Lem. 2}}{\forall M,N \in P_{in} : \tau(N) \vdash \tau(M) \Rightarrow output_\alpha(\tau(M)) \subseteq output_\alpha(\tau(N))}\text{Lem. 1}}{\forall a,b \in L_{in} : b \vdash a \Rightarrow output_\alpha(a) \subseteq output_\alpha(b)}\text{Def. 6}}{\forall a,b \in L_{in} : b \vdash a \Rightarrow \forall x \in L : output_\alpha(a,x) \Rightarrow output_\alpha(b,x)}\text{Def. 6}$$

Secondly, it satisfies the disjunction rule (OR) for input formulas:

$$\dfrac{\dfrac{\dfrac{\forall M \in P_{in} : gci(\{Out(\alpha(N)) \mid N \in W_{in} \wedge M \leq N\} \leq Out(\alpha(M)))}{\forall M \in P_{in} \forall p \in InSig(\Delta) : \tau(Out(\alpha(M)) \vdash output_\alpha(\tau(M) \wedge p) \vee output_\alpha(\tau(M) \wedge \neg p))}\text{Lem. 6}}{\forall a \in P_{in} \forall b \in InSig(\Delta) : output_\alpha(a) \subseteq output_\alpha(a \wedge b) \cap output_\alpha(a \wedge \neg b)}\text{Def. 6}}{\forall a \in P_{in} \forall b \in InSig(\Delta) : \forall x \in L : (a \wedge b, x) \in output_\alpha, (a \wedge \neg b, x) \in output_\alpha \Rightarrow (a,x) \in output_\alpha}\text{Def. 6}$$

■

Finally we relate functionality description α to the set of generators G, that is, we define a mapping π from functionality descriptions to sets of generators such that $output_\alpha = output(\pi(\alpha))$. In the general case, we can simply define $G = \{(\tau(M), \tau(N)) \mid M \in P_{in}, N = Out(\alpha(M))\}$, but for functionality descriptions satisfying additional properties, more concise representations for the set of generators can be given.

THEOREM 3. *If* $G = \{(\tau(M), \tau(N)) \mid M \in P_{in}, N = Out(\alpha(M))\}$, *then* $output_\alpha = output(\pi(\alpha))$.

The logical characterization of the properties of functionality descriptions suggests various other properties of functionality descriptions that can be studied. For example, a notion of well-informed can be defined as simply the validity of the OR rule, and a notion of self-boundedness can be defined using CT instead of Trans. Moreover, the logical characterization may be used to extend the notion of functionality description to more expressive languages, for example, languages that include disjunction. We do not further consider these issues in this article.

Summarizing, Theorem 2 relates compositional design to cognitive attitudes, in the sense that typical examples of input/output processes studied in input/output logic arise in contexts of conditional beliefs, obligations, goals, ideals, preferences, and actions. This is relevant for the application of compositional design of agent architectures such as the BOID architecture considered in this article. The major issue for further research for the foundations of compositional architectures is the logical analysis of the *interaction* of input/output components,[17] such as the distinction between object and meta-level interactions (for which hierarchical logics may be used) and feedback loops among components (for which generalized self-boundedness or generalized CT rules may be used). A first step toward such issues is discussed in the following section.

4. COORDINATION OF COMPONENTS

In this section we discuss the coordination of components. In Section 4.1 we discuss how functionality descriptions of individual components can be combined, and in Section 4.2 we discuss their formalization in logic.

4.1. Composition Relations . . .

Functionality descriptions can be combined by composing functions in the usual way. For example, consider the goal-based agent in Figure 1 with input F (observations), and two functionality descriptions, α_p (plan generation) and α_g (goal generation). They can be combined to β by first applying α_g on F and thereafter α_p, that is, $\beta(F) = \alpha_p(\alpha_g(F))$ or

$$\beta(F) = \alpha_p \circ \alpha_g(F)$$

Moreover, consider the more complex example of the goal generation component in Figure 2. As a first naive approximation, any subset of the four components can be applied in any order, that is

$$\alpha_g = \alpha_b \cup \alpha_o \cup \cdots \cup \alpha_b \circ \alpha_o \cup \cdots \cup \alpha_b \circ \alpha_o \circ \alpha_i \circ \alpha_d$$

This is obviously too strong, but in another sense, it is also too weak.

A first reason for calling this composition too strong is that it connects every component to every other component. This is not in accordance with, for instance,

the architecture as shown in Figure 2; the input of the belief component contains the observations as well as the output of the obligation, intention, and desire components. Moreover, the output of the belief component is itself connected to the input of the obligation, intention, and desire components. As explained in Section 2, this means that the motivational attitudes induce new beliefs, and that the beliefs may induce new motivational attitudes, respectively. Which beliefs and motivational attitudes are derived depends on the rules and the properties of the components, that is, on their input/output logic. The behavior of the BOID component depends on rules of the individual components, properties of the individual components (i.e., their input/output logic), and the links between components (which are assumed to be fixed as in Figure 2). The most interesting property is the identity or ID rule, which states that the input is included in the output. For some components this may lead to desired results, but for other components this leads to counterintuitive results. For example, in the BOID architecture it may make sense for beliefs to include the observations, but it makes less sense for obligations or desires to include the beliefs. You may desire to go to the dentist and believe that going to the dentist implies pain, but this does not imply the desire for pain.[5] We can implement these differences in input–output behavior simply by accepting the identity rule for the belief component without accepting it for the obligation, intention, and desire components.

A second reason for calling the composition too strong is that it neglects the control loop, and in particular the conflict resolution mechanisms. For example, Thomason[6] argues that beliefs should override desires with the following example. If you think it is going to rain and you believe that if it rains, you will get wet, and you would not like to get wet, then you have to conclude that you get wet. Beliefs should therefore prevail in conflicts with desires. This issue is further discussed in the following section.

But, the composition is also too weak, because it does not take complex mechanisms like feedback loops and reasoning by cases into account. These are exactly the properties discussed in Section 3 of self-boundedness and well informedness, that is, of the cumulative transitivity (CT) and disjunction (OR) rule. The CT rule, or its variant the transitivity (TRANS) rule, is valid within the belief component (rules can be applied one after the other), and it is valid among the components via the feedback loops. The OR rule states that the component satisfies reasoning by cases. For example, suppose there are two belief rules:

- If a, then the agent believes x;
- If $\neg a$, then the agent believes x.

Can we conclude that the agent believes x if there are no observations? We can if the agent's belief component supports reasoning by cases. It is usually considered to be a reasonable conclusion, but is seldom implemented due to the complexity of generating cases. A more complex variant of this reasoning scheme among components is, for example, the following two rules:

- If a, then the agent desires x;
- If $\neg a$, then the agent is obliged x.

Can we conclude that the agent is motivated for x if there are no observations or beliefs? Only if the agent's control loop supports reasoning by cases.

4.2. … and Global Constraints

The general approach we advocate to combine functionality specifications is

(1) to define a very general composition that allows all possible behaviors of the system such that it is not too weak, and thereafter
(2) add constraints on this composition such that it is not too strong

The first step of this approach is usually straightforward (e.g., introduce some operators for iteration), but the second may involve many different techniques. For example, a constraint that can be added on the composition is the realism constraint. One way to express this constraint is to construct a conflict resolution mechanism in which beliefs override desires, such as the mechanisms proposed in Thomason's BDP logic[6] and in Broersen et al.'s BOID architecture.[9] An alternative, more general, and more insightful approach is to formulate the intuition as a property of the functional agent description, that is, as the relation between observations and extensions.[7] Consider the following conflict:

(1) The agent believes the car will be sold.
(2) The agent believes the car will not be sold.

The agent does not know what to believe: It is confused. Alternatively, the agent has two incompatible belief sets, one that argues that the car will be sold and another that argues that the car will not be sold. Confusion can be formalized as an inconsistent belief set, whereas multiple belief states can be formalized by multiple extensions in, for example, Reiter's default logic.[18] In this article we follow the latter approach. Consider the following conflict:

(1) The agent desires the car to be sold.
(2) The agent desires the car not to be sold.

The agent has two conflicting desires, which may both become candidate goals. We call this an internal desire conflict. In this article we assume that also a conflict among desires leads to multiple extensions. Finally, consider the following conflict:

(1) The agent believes the car will be sold.
(2) The agent desires the car not to be sold.

The agent's desire conflicts with its belief. Such mixed conflicts can be interpreted in various ways. One way, which we adopt in this article, is due to Thomason.[6] He argues that it is unrealistic to allow the agent's desire to become a goal when it conflicts with its beliefs, and that therefore beliefs should override desires,

with the following example. If the agent believes it is raining and it believes that if it rains, it will get wet, and it desires not to get wet, then the agent cannot pursue the goal of not getting wet. This example shows that it is unrealistic (i.e., it allows for wishful thinking) to allow the desire of not getting wet to become a goal. Thus, beliefs prevail in conflicts with desires. Thomason's interpretation can be contrasted with the following example:

(1) The agent believes the fence is white.
(2) The agent desires the fence to be green.

In this example the agent can see to it that the fence becomes green by painting it, so pursuing the goal that the fence is green is not wishful thinking. The difference between this example and the previous one is that this is not a conflict due to implicit temporal references. The belief implicitly refers to the present whereas the desire refers to the future:

(1) The agent believes the fence is white *now*.
(2) The agent desires the fence to be green *in the future*.

In this article we only use abstract examples in which we do not give an interpretation for the propositional atoms. If there is a conflict between a belief and a desire, then there is always a real conflict (as in the car selling example), and never an apparent conflict (as in the fence example) that can be solved by considering time. We also do not discuss the kind of revision or updating involved in the fence example.

An intriguing question is how to resolve conflicts among beliefs and desires, in case more than two rules are involved. Consider the example in Figure 3.

Figure 3 represents the following four rules:

(1) The agent believes p.
(2) The agent desires q.
(3) If p, then the agent desires r.
(4) If q, then the agent believes $\neg r$.

In the sequel we give some definitions to determine whether it is realistic to desire q or r.

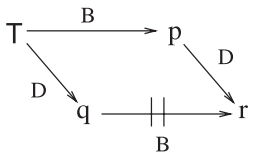

Figure 3. Is it realistic to desire q? Is it realistic to desire r?

We introduce two properties characterizing realistic desires. They are not restricted to one particular logic or architecture, but they can be applied to any extension-based approach.

The agent is characterized by a function, which we denote by Δ, from so-called BD theories (sets of belief and desire rules) to extensions (logically closed sets of propositional sentences). This terminology is inspired by Reiter's default logic.[18] However, for now we do not assume any further properties on the relation between BD theories and their extensions. For example, we do not assume that observations are included in the extensions. In this sense the phrase "extension" may be slightly misleading.

DEFINITION 7 (Agent, BD Theory, Extension). *A BD theory is a tuple* $T = \langle W, B, D\rangle$*, where* W *is a set of propositional sentences of a propositional language* L *and* B *and* D *sets of ordered pairs of such sentences. An extension is a logically closed set of* L *sentences.* Δ *is a function that returns for each BD theory a set of extensions. We write* $\Delta(T)$ *for the set of all extensions of a BD theory* T *(there may be none, one, or multiple extensions), and we write* $Th_L(S)$ *for all propositional consequences of the set of propositional formulas* S*. For representational convenience we write* $\Delta(W, B, D)$ *for* $\Delta(\langle W, B, D\rangle)$*, and we write* $Th_L(\alpha)$ *for* $Th_L(\{\alpha\})$*.*

Using Definition 1, the example of Figure 3 can be represented by a BD theory $T = \langle \emptyset, \{\top \overset{B}{\hookrightarrow} p, q \overset{B}{\hookrightarrow} \neg r\}, \{\top \overset{D}{\hookrightarrow} q, p \overset{D}{\hookrightarrow} r\}\rangle$. Note that this definition allows us to use any pairs of propositional formulas, which means that we consider a more general setting than in the examples thus far.

Realism concerns the rational balance in case of conflict. Therefore we first define what a conflict is. Conflicting theories lead to an inconsistent extension if rules are applied.[e]

DEFINITION 8 (Conflict). *Let* $T = \langle W, B, D\rangle$ *be a BD theory.* T *contains a conflict iff there is no consistent set* E *of* L *sentences such that*

- $W \subseteq E$.
- *If* $a \overset{B}{\hookrightarrow} x \in B$ *or* $a \overset{D}{\hookrightarrow} x \in D$ *and* $a \in Th_L(E)$*, then* $x \in Th_L(E)$.

One way to proceed is to define for each BD theory when a desire is realistic and when it is unrealistic. A drawback of this approach is that it has to commit to a logic of rules for the belief and desire rules. We therefore follow another approach, which may be called comparative. The basic pattern is as follows. If a realistic

[e] An alternative stronger definition is that T is a conflict if the following is inconsistent:

$$W \cup \{a \rightarrow x \mid a \overset{B}{\hookrightarrow} x \in B \text{ or } a \overset{D}{\hookrightarrow} x \in D\}$$

Which definition of conflict is used depends on the underlying logic of rules; see, for example, Ref. 3 for some possibilities. For the definitions of realism in this article the exact definition of conflict is not important.

function Δ returns for a BD theory T a set of extensions S, then we can deduce that it does not return for other BD theories T' extensions S'. The latter extensions S' would be unrealistic, that is, based on unrealistic desires.

The realism properties are defined in terms of sets of belief and desire rules. However, BD theories are not mapped on a set of belief and desire rules, but on extensions generated by such rules. We therefore have to associate with each extension a set of belief and desire rules. The following definition associates an extension with the set of rules that are applied in it (sometimes called its generating set[18]).

DEFINITION 9 (Applied Rules). *Let $T = \langle W, B, D\rangle$ be a BD theory and let the set E be one of its extensions. The set of applied belief rules in extension E is $R_B(T,E) = \{\alpha \overset{B}{\hookrightarrow} w \in B \mid \alpha \wedge w \in E\}$, and the set of applied desire rules is $R_D(T,E) = \{\alpha \overset{D}{\hookrightarrow} w \in D \mid \alpha \wedge w \in E\}$.*

The intuition behind a priori realism in Definition 10 is as follows. Consider a BD theory ($\langle W, B, D\rangle$) and an extension of this BD theory (E) in which at least one desire has been applied. We call this the a posteriori state. We want to ensure that these applied desires result in realistic extensions. We therefore consider the state in which this rule has not been applied (BD theory $\langle W, B, D'\rangle$ with extension E'). We call this state the a priori state. We now say that the desire rule is realistic if the set of applied belief rules in the a priori state is a subset of the set of applied belief rules in the a posteriori state. This implies that the removal of realistic desires from the BD theory can only decrease the extension, not increase it or remove it.[f]

DEFINITION 10 (A Priori Realism). *Δ is a priori realistic iff for each $E \in \Delta(W,B,D)$ and $D' \subseteq R_D(\langle W,B,D\rangle, E)$ there is an $E' \in \Delta(W,B,D')$ such that we have $R_B(\langle W,B,D'\rangle, E') \subseteq R_B(\langle W,B,D\rangle, E)$. We also say that each $E \in \Delta(W,B,D)$ that satisfies the above condition is realistic, and we say that all applied desire rules of a realistic extension are realistic desire rules.*

In the remainder of this section we illustrate a priori realism by some examples.

Example 1. Consider the four triangles in Figure 4. Intuitively, we have case a, $\{p,q\}$; case b, $\{p\}$ or $\{p,q\}$; case c, $\{\neg p,q\}$ or $\{p,q\}$; and case d, $\{p\}$ or $\{\neg p,q\}$.

$$\text{Case a. Let } T = \langle \varnothing, \{\top \overset{B}{\hookrightarrow} p,\ \top \overset{B}{\hookrightarrow} q\}, \{q \overset{D}{\hookrightarrow} \neg p\}\rangle \text{ and}$$

$$T' = \langle \varnothing, \{\top \overset{B}{\hookrightarrow} p,\ \top \overset{B}{\hookrightarrow} q\}, \varnothing\rangle$$

[f]An alternative, closely related definition of a priori realism is as follows. For each $E \in \Delta(W,B,D)$ and $D' \subseteq R_D(\langle W,B,D\rangle, E)$ there is an $E' \in \Delta(W,B,D')$ such that $E' \subseteq E$. A simple instance of this property, which we may call "restricted a priori realism," is the case where D is the empty set. This property says that every BD extension extends a B extension. For each $E \in \Delta(W,B,D)$ there is an $E' \in \Delta(W,B,\varnothing)$ such that $E' \subseteq E$.

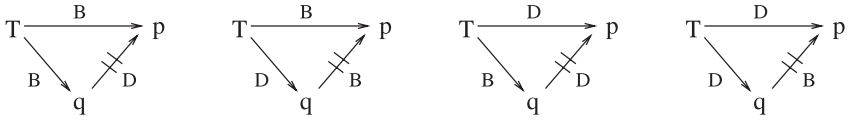

Figure 4. Desire–belief triangles.

T contains a conflict, because any set E as defined in Definition 8 contains p as well as $\neg p$. Moreover, assume $\Delta(T') = \{Th_L(p \wedge q)\}$ with $R_B(T', Th_L(p \wedge q)) = \{\top \overset{B}{\hookrightarrow} p, \top \overset{B}{\hookrightarrow} q\}$. Due to a priori realism we have for each element E of $\Delta(T)$ that $R_B(T, E)$ contains $\{\top \overset{B}{\hookrightarrow} p, \top \overset{B}{\hookrightarrow} q\}$, and consequently E has to contain $Th_L(p \wedge q)$. $\Delta(T)$ thus cannot contain, for example, $Th_L(q \wedge \neg p)$. In other words, according to Property 10 we have that the desire for $\neg p$ is unrealistic.

Case b. Let $T = \langle \emptyset, \{\top \overset{B}{\hookrightarrow} p, q \overset{B}{\hookrightarrow} \neg p\}, \{\top \overset{D}{\hookrightarrow} q\}\rangle$ and

$$T' = \langle \emptyset, \{\top \overset{B}{\hookrightarrow} p, q \overset{B}{\hookrightarrow} \neg p\}, \emptyset\rangle$$

T contains a conflict, because any set E as defined in Definition 8 contains p as well as $\neg p$. Assume $\Delta(T') = \{Th_L(p)\}$ with $R_B(T', Th_L(p)) = \{\top \overset{B}{\hookrightarrow} p\}$. Due to a priori realism, each element of $\Delta(T)$ has to contain $Th_L(p)$. Consequently, $\Delta(T)$ thus can contain $Th_L(p \wedge q)$, but it cannot contain, for example, $Th_L(q \wedge \neg p)$. In other words, according to Property 10 we have that $\neg p$ is unrealistic. Note that there is not a desire for $\neg p$, but that $\neg p$ would be a believed consequence of a desire (for q). However, it does not imply that the desire for q is unrealistic.

Case c. Let $T = \langle \emptyset, \{\top \overset{B}{\hookrightarrow} q\}, \{\top \overset{D}{\hookrightarrow} p, q \overset{D}{\hookrightarrow} \neg p\}\rangle$ and

$$T' = \langle \emptyset, \{\top \overset{B}{\hookrightarrow} q\}, \emptyset\rangle$$

If $\Delta(T') = \{Th_L(q)\}$, then each element of $\Delta(T)$ has to contain $Th_L(q)$, but $\Delta(T)$ still can contain, for example, $Th_L(p \wedge q)$ and $Th_L(\neg p \wedge q)$. In other words, according to Property 10 in T' neither p nor $\neg p$ would be unrealistic. It illustrates that Property 10 does not classify conflicts among desires as unrealistic.

Case d. Let $T = \langle \emptyset, \{q \overset{B}{\hookrightarrow} \neg p\}, \{\top \overset{D}{\hookrightarrow} p, \top \overset{D}{\hookrightarrow} q\}\rangle$,

$$T' = \langle \emptyset, \{q \overset{B}{\hookrightarrow} \neg p\}, \{\top \overset{D}{\hookrightarrow} p\}\rangle, \text{ and}$$

$$T'' = \langle \emptyset, \{q \overset{B}{\hookrightarrow} \neg p\}, \{\top \overset{D}{\hookrightarrow} q\}\rangle$$

If $\Delta(T') = \{Th_L(p)\}$ and $\Delta(T'') = \{Th_L(q, \neg p)\}$, then a priori realism implies $Th_L(p \wedge q) \notin \Delta(T)$. However, note that $Th_L(p)$ and $Th_L(q \wedge \neg p)$ may be in $\Delta(T)$ (verification left to the reader).

5. FURTHER RESEARCH

There are several consequences of Theorem 2 for further logical investigations.

First, the declarative functionality descriptions can be compared to other logical formalisms. For example, out_3^+ is the logic of Reiter's normal default logic without constraints, and out_4^+ is material implication (and thus satisfies cut elimination, in the sense that it is equivalent to out_2^+).[3] Moreover, some lemmas shown for the mapping α now correspond to well-known logical properties. For example, the fact that α is monotonic if it is conservative and self-bounded now corresponds to the fact that Id and TRANS imply SI (in the context of WO), proven as follows.

$$\dfrac{\dfrac{\dfrac{-}{(a \wedge b, a \wedge b)}\,\text{ID}}{(a \wedge b, a)}\,\text{WO} \qquad (a, x)}{(a \wedge b, x)}\,\text{TRANS}$$

Second, the declarative functionality descriptions can be compared to other possible mappings. A possible extension is to allow for defeasible inference. Systems without SI have been studied in conditional logic[19] and more recently in Ref. 20: Id + CT is cumulative inference, Id + CT + OR is preferential inference.

Third, with the partitioning of the signature, we can use weaker logical systems to obtain the same output. We do not need the Id rule if the input and output variables do not overlap; that is, we can use out_3 and out_4 instead of out_3^+ and out_4^+. Moreover, if we also do not have intermediate variables then we can drop CT, that is, we can use out_1 and out_2. This shows the use of all these different systems: The more complex a combination of variables is allowed, the more an extended proof system (input/output logic) is required.

Fourth, another issue of relevance for complex compositional systems is their computational complexity. Note, for instance, that an extended set of proof rules does not necessarily mean increased computational complexity (e.g., out_2 and out_3 are probably computationally more complex than out_1 or out_4^+). Again the logical characterization may be of use, because complexity results may already be known or such results are easier to obtain in a standard logic setting (thus far no complexity results on input/output logics are known, but results are known on instances like Reiter's normal default logic).

Moreover, in this article we have just made a start with the formal analysis of the coordination of cognitive components. There are many issues to be studied.

6. RELATED RESEARCH

In agent technology, the relation between architectures and formal logic is a topic of frequent discussion. The most discussed example is the relation between Georgeff and Rao's BDI architectures[21–23] and their BDI logic.[12] Typical questions are, "How do we use formal languages in the design of such architectures?" "What is the relation between the architecture and its logic?" and more generally, "How do we exploit computational logic for multiagent systems?" There are at least two

fundamentally different uses of logics for agents. Languages for analysis, requirement specification, and verification enable reasoning *about* agents, and knowledge specification languages are used *within* agent architecture to represent agent's mental attitudes. Despite this clear distinction in the *uses* of agent logics, there is no such clear distinction in the agent logics themselves, because some logics can be used as specification as well as knowledge representation languages. An example is the modal BDICTL logic[12,24]—though it seems better fit for the former task.

In practical reasoning, monadic modal logics were introduced to formalize various cognitive notions, most notably epistemic operators (knowledge and belief) and deontic operators (obligation and permission). However, the use of monadic modalities was soon seen as too restrictive. The area of epistemic logic transformed into the area of belief revision, in particular under the Alchourrón–Gärdenfors–Makinson paradigm.[14] The area of deontic logic studied an issue related to belief revision, which we may call obligation revision (though it is usually called contrary-to-duty reasoning[25]). Von Wright,[26] Alchourrón and Bulygin,[27] Alchourrón,[28] and Makinson[29] started to look for general characterizations of the problems, and they concluded that modal logics with their focus on nested modalities did not touch the core of the problem, which was to be found in the notion of conditionality. Input/output logic, as used in this article, is one of the logics developed in this new tradition. Moreover, several other dynamic approaches have been introduced lately in the area of logic, in particular in relation to language and philosophy, such as Veltman's update semantics[30] and van Benthem's work[31] on logical dynamics.

In general, previous theoretical research has often neglected to show possible implementations. As in Refs. 8 and 32, we not only provide a theoretical framework, but we also provide an architecture that includes a control procedure in the style of Rao and Georgeff's Procedural Reasoning System (PRS).[21,22] In particular, we provide a theoretical framework to specify an agent's properties by extending the BDP formalism with prior intentions and obligations, and use BOID as a generic agent architecture to design and implement agents. In this section, we discuss three existing formalisms and explain how our formalism is related to them.

6.1. Deontic Action Programs

Deontic action programs, introduced by Eiter et al.,[10] are sets of Horn-type rules of the form $L_1, L_2, \ldots L_n \rightarrow A$, where the L_i are literals (possibly negated atoms), either of a Boolean form or from the set $P(\alpha), F(\alpha), O(\alpha), W(\alpha), Do(\alpha)$, which stand for permission, prohibition, obligation, waiving, and performance of α, respectively. In the head A, negations and Boolean combinations are not allowed. The "semantics" of rules is defined in terms of status sets, which are application sets of the above operators to ground instances of actions α. The status sets correspond to what we call "extensions." Eiter et al. discuss many different choices for calculating with action programs, resulting in many different types of status sets. They propose closure of status sets S under the deontic rules:

If $O(\alpha) \in S$, then $Do(\alpha) \in S$.
If $Do(\alpha) \in S$, then $P(\alpha) \in S$.

Furthermore, status sets have to obey consistency constraints:

If $O(\alpha) \in S$, then $W(\alpha) \notin S$.
If $P(\alpha) \in S$, then $F(\alpha) \notin S$.
If $P(\alpha) \in S$, then $Pos(\alpha)$ (a rule ensuring that α is only permitted if its precondition is satisfied).

Status sets closed under the program rules, the above deontic rules, obeying the consistency constraints and other, independently specified constraints, are called "feasible status sets." But feasible status sets lack a criterion of minimality: There may be unsupported elements. This leads to the concepts of "groundedness" and "rational status set." The class of rational status sets is a subset of the class of feasible status sets. The criterion that it should not be possible to use rules contrapositively leads to the concept of "reasonable status set," the class of reasonable status sets being a subset of the class of rational status sets.

Deontic action programs and BOID theories have several aspects in common. Both formalizations are rule based, encompass normative concepts (obligation), and do not allow for contrapositive rule application. But there are also many notable differences. A technical difference follows from the comparison with Reiter's default theories. BOID rules are like normal defaults: Information in the heads of rules is allowed to occur negated and may give rise to conflicts, whereas negation symbols in the body are interpreted in the standard way. In the rules of action programs, negations in the heads are not allowed, whereas negation in the bodies is interpreted according to "negation as failure." Also, both approaches have different subject domains: BOID rules do not deal with permission, prohibition, waiving, and executability, whereas action programs do not deal with beliefs, desires, and intentions. From this difference in subject domain follows a difference in research questions. The central problem for BOID theories is conflict resolution. BOID aims at selecting actions for execution, given possibly conflicting information about informational, internal, and external motivational attitudes. Action programs, on the other hand, aim at selecting actions for execution, given information about action and external motivational attitudes. Possible conflicts with internal or informational attitudes are not considered. In the action selection process, BOID always comes to an answer, whereas action programs may return the empty action.

From a deontic perspective, it is possible to raise some objections against the approach taken by Eiter et al. A first observation is that status sets fail to establish the important deontic relation that in standard deontic logic (SDL[33]) is expressed as $O(\phi) \leftrightarrow F(\neg\phi)$, and that in deontic action logics is expressed as $O(\alpha) \leftrightarrow F(\sim\alpha)$ (see Ref. 34). In other words, this property says that "obligation to do α" equals "prohibition to do anything other than α." If Eiter et al. want to be faithful to this deontic principle, they should consider incorporation of the rules: (1) if $O(\alpha) \in S$, then $F(\beta) \in S$ for all actions β other than α, and (2) if $F(\beta) \in S$ for all actions β other than α, then $O(\alpha) \in S$. Especially this second clause might be important, because it may lead to selection of actions for execution only from prohibitions for "alternative" actions.

A more fundamental objection concerns the motivation for including deontic operators in action programs in the first place. The main reason for viewing deontic notions as useful for modeling behavior is their capacity to specify violations.[35] In the use Eiter et al. make of deontic operators, such violations do not occur. The central elements of status sets are the elements $Do(\alpha)$ that determine the action that is undertaken. The other deontic elements of status sets have no independent operational meaning in themselves and are only important in the sense that they impose restrictions on occurrences of elements $Do(\alpha)$. This means that no violations of deontic constraints are considered: Status sets always obey all specified norms. The authors notice this weakness themselves and suggest that it can be remedied by allowing and deciding internal conflicts among obligations and by the introduction of violation costs. But this does not solve the problem of how to specify the way an agent is supposed to behave when a certain norm violation has occurred (see Ref. 34).

6.2. BDI Approach

The BDI approach[12,21–23] is one of the most popular approaches to capturing the rational balance between mental attitudes of cognitive agents. In the BDI framework, beliefs, desires, and intentions are considered as independent modalities formalized by separate modal operators. The specific relationships among these modalities are given by two types of constraints called static and dynamic constraints. The static constraints are formalized by special axioms, which, for example, ensure stable decision-making behavior. The dynamic constraints, known as *commitment strategies*, capture the dynamic of mental attitudes, that is, the ways in which the (relations between) attitudes change over time. An important commitment strategy concerns the perseverance with which an agent maintains its goals. In Rao and Georgeff's formalism $Int_a(A\Diamond\alpha)$ describes the pursuit of a goal: Agent a intends to eventually achieve α, no matter how the future evolves. The condition that a proposition α will eventually come true is expressed by $\Diamond\alpha$, and A is to be read as "for all possible courses of future events." The question here is how long an agent should maintain $Int_a(A\Diamond\alpha)$. It seems to be reasonable for an agent to maintain a goal α as long as the agent believes that α is attainable. This is exactly what $Bel_a(\alpha)$ denotes. The agent should then give up to maintain a goal as soon as the agent believes that α is not attainable anymore. This condition can be expressed with the help of the E operator, which reads "for at least one possible course of events." This condition is translated into $\neg Bel_a(E\Diamond\alpha)$. The until operator U then allows us to formulate an axiom of change:

$$A(Int_a(A\Diamond\alpha)U(Bel_a(\alpha) \vee \neg Bel_a(E\Diamond\alpha)))$$

This is what Rao and Georgeff call "single-minded commitment." Other commitment strategies are defined in a similar way. Commitment strategies of this type are used to characterize different types of agents.

Like BDI, BOID aims at a formalization of the rational balance between mental attitudes of agents. However, the BOID formalism extends the mental attitudes

of BDI with obligations. The gap between BOID architectures and their underlying logics is much smaller than the large gap between modal BDI logics and their architectures. This gap is due to the fact that the BDI formalism is developed to *specify* agent behavior whereas the BOID formalism is developed to *design* agents. Input/output logic can employ the BOID formalism to specify agent behavior in a straightforward way. In fact, in this article we have shown how input/output logic can be used to specify the behavior of individual components.

Finally, in modal logics, *nested* modalities can be represented, such as: "You are obliged not to desire to smoke." This cannot be represented in input/output logics or in Reiter's default logic. However, we can replace the propositional base language of input/output logics by a modal language containing B/O/I/D modalities. Now rules may be added such that the desired extensions can be derived that can be used as goals in the BOID architecture.[36]

6.3. BDP Approach

Another formalism to specify the behavior of cognitive agents is known as the Beliefs, Desire, and Planning (BDP) formalism proposed by Thomason.[6] BDP is designed to integrate reasoning about beliefs and desires with planning. The basic idea is to extend the planning formalisms with goals that are not given but derived from an agent's beliefs and desires. Thomason first discusses the BD formalism without planning and only thereafter BDP, such that in the first part he can focus on the interaction between beliefs and desires.

The BDP formalism is based on Reiter's default logic.[18] The basic idea is to model beliefs and desires both as Reiter defaults, *without modalities for belief or desire*, such that the extensions contain all the indirect effects of actions. That is, a BD basis is a tuple $\langle M, NB, ND \rangle$ with M a set of formulas, NB a set of B-rules "if a, then I belief x" written as $a \overset{B}{\hookrightarrow} x$, and ND a set of D-rules "if a, then I desire x" written as $a \overset{D}{\hookrightarrow} x$. Extensions are built in the usual way without distinguishing between beliefs and desires, so, for example, the BD basis $a, a \overset{B}{\hookrightarrow} b, b \overset{D}{\hookrightarrow} c$ has as an extension $Th(\{a, b, c\})$. But then, there are two types of conflicts:

- Conflicts between a belief and a desire lead to the overriding of desire by belief to block wishful thinking.
- Other conflicts such as conflicts between two desires lead to multiple extensions.

BDP is partly inspired by computational BDI systems, but there are important differences. First, in BDI logics decisions are restrained by previous intentions. Second, whereas BDI logics only formalize unconditional beliefs and desires, BDP also formalizes conditional ones. Third, whereas BDI logics are based on modal logic, BD logic (a subformalism of the BDP logic) is based on Reiter's normal default logic.

The BOID formalism extends the BDP with Obligations and Intentions. In particular, the BDP formalism is based on conflict resolution for conditional beliefs and desires that is extended in the BOID formalism with conditional obligations and desires borrowed from, respectively, deontic action programs[10] and BDI

logic.[11,12] However, in contrast to the BDP formalism, the BOID formalism does not account for planning, though this is not a principal shortcoming. We believe that the computed and selected extension in the BOID formalism, which can be considered as a set of goals, should be mapped into a sequence of actions (i.e., a plan) through which the goals can be reached. The mapping from a set of goals to a plan can be accomplished by an additional planning component in the BOID architecture.[36]

Extending the BDP logic with obligations and intentions increases the number of possible conflicts dramatically. In both approaches conflict resolution is defined as selecting a subset of the conditionals that does not derive a contradiction. As different types of rules are formalized as normal Reiter defaults, this means that a Reiter extension should be selected. The fact that extensions are built represents that all the effects of the decisions are taken into account.

To design cognitive agents based on the BDP formalism, Thomason sketches an agent architecture[6] that is far from being used to design cognitive agents. For this reason, we have introduced the BOID architecture,[9] which can be used to design cognitive agents based on an extension of the BDP formalism.

7. CONCLUDING REMARKS

To scale up agent applications from toy examples to industrial applications, agent logics need to support compositional or component-based design and include related tools like abstraction and refinement. In compositional architectures for reasoning systems or agents, the functioning of the overall system is based on the functionality of the components, in addition to the composition relations by which they are glued together within the compositional architecture. Specification of a composition relation may involve, for example, the possibilities for information exchange among components and control.

In this article we consider the compositional design of the cognitive BOID architecture. In a component-based BOID agent, questions concerning the behavior of the whole agent or satisfaction of the overall agent's functionality can be decomposed into questions for the case of a single primitive interactive reasoning component within the agent and questions related to the agent's compositional structure. For the former type of questions, the notion of declarative functionality description comes into play. At the level of the composition as a whole, the (implementation) internal details of a component are irrelevant, as long as an adequate overall description of the component's behavior is available. Because functionality descriptions abstract from these internal details, this notion is well suited for these purposes. As cognitive attitudes are represented by components, the cognitive attitudes have to represented by conditional structures—for which we used input/output logics. Thus, compositionality forces us to use a more complex logic to describe the behavior of components as well as the composition of components.

For the specification of individual components, we relate functionalities of reasoning components to input/output logic,[3,4] in the sense that it has been shown how the formalization of the functionality description of a reasoning component, introduced in Ref. 2, can be mapped onto an input–output logic specification, and,

moreover, how specific properties of such functionality descriptions correspond to specific proof rules in input–output logic. This introduces a structural relation among components within architectures of reasoning systems, as specified from a software engineering perspective (e.g., in a design language as DESIRE) to logic. It shows that input–output logics can be used as the basis of an adequate formalization of the semantics of a component in a compositional agent-based reasoning system.

Compositionality has not only some important advantages for design, but it also may be a crucial factor for formal analysis like verification. This is a topic for further research. Compositional verification extends the main advantages of compositionality for designing to:

A well-structured verification process. Both conceptually and computationally the complexity of the verification process can be handled by compositionality at different levels of abstraction. The verification problem is reduced to a number of much simpler verification problems, and the problem of defining the logical relation that compose properties of components together to properties of the whole system.

The reusability of proofs for properties of reused components. If a modified component satisfies the same properties as the previous one, then the proof of the properties at the higher levels of abstraction can be reused to show that the new system has the same properties as the original. This has high value for a library of reusable generic models and components. A library of reusable components and task models may consist of both specifications of the components and models and their design rationale. As part of the design rationale, at least the properties of the components and their logical relations can be documented. For more details about compositional verification, see Refs. 37–39.

Acknowledgments

Thanks to Zhisheng Huang, Joris Hulstijn, and Jan Treur for discussions on the issues discussed in this article.

References

1. Brazier FMT, Jonker CM, Treur J. Principles of compositional multi-agent system development. In: Cuena J, editor. Proc 15th IFIP World Computer Congress, WCC'98, Conference on Information Technology and Knowledge Systems, IT&KNOWS'98. Amsterdam: IOS Press; 1998. pp 347–360.
2. Treur J. Semantic formalisation of interactive reasoning functionality. Int J Intell Syst 2002;17:645–686.
3. Makinson D, van der Torre L. Input–output logics. J Philos Logic 2000;29:383–408.
4. Makinson D, van der Torre L. Constraints for input–output logics. J Philos Logic 2001;30:155–185.
5. Bratman ME. Intention, plans, and practical reason. Cambridge, MA: Harvard University Press; 1987.

6. Thomason R. Desires and defaults. In: Proc Seventh Int Conf on Principles of Knowledge Representation and Reasoning (KR'2000). San Francisco, CA: Morgan Kaufmann; 2000. pp 702–713.
7. Broersen J, Dastani M, van der Torre L. Realistic desires. J Appl Non-Class Logic 2002;12:287–308.
8. Castelfranchi C, Dignum F, Jonker C, Treur J. Deliberative normative agents: Principles and architecture. In: Proc Agent Theories, Architectures, and Languages, Lecture Notes in Computer Science. Berlin: Springer-Verlag; 1999. pp 364–378.
9. Broersen J, Dastani M, Hulstijn J, van der Torre L. Goal generation in the BOID architecture. Cognit Sci Q 2002;2:428–447.
10. Eiter T, Subrahmanian VS, Pick G. Heterogeneous active agents I: Semantics. Artif Intell 1999;108:179–255.
11. Cohen PR, Levesque HJ. Intention is choice with commitment. Artif Intell 1990;42: 213–261.
12. Rao A, Georgeff M. Modeling rational agents within a BDI architecture. In: Proc Int Conf on Knowledge Representation and Reasoning (KR'91). San Francisco: Morgan Kaufmann; 1991. pp 473–484.
13. Dastani M, van der Torre L. What is a normative goal? Towards goal-based normative agent architectures. In: Regulated Agent-Based Systems. Postproceedings International Workshop on Regulated Agent-Based Social Systems: Theories and Applications (RASTA02), Lecture Notes in Artificial Intelligence 2934, Berlin: Springer; 2004. pp 210–227.
14. Alchourrón CE, Gärdenfors P, Makinson D. On the logic of theory change: Partial meet contraction and revision functions. J Symbolic Logic 1985;50:510–530.
15. Arbab F. Abstract behavior types: A foundation model for components and their composition. In: Formal Methods for Components and Objects: First International Symposium, FMCO 2002, Leiden, The Netherlands, November 5–8, 2002, Revised Lectures. Lecture Notes in Computer Science 2852. Berlin: Springer-Verlag; 2003. pp 33–70.
16. The BOID's home page. http://boid.info.
17. Leemans P, Treur J, Willems M. On the verification of knowledge-based reasoning modules. Technical Report IR-346, Vrije universiteit, faculteit der wiskunde en informatica; 1993.
18. Reiter R. A logic for default reasoning. Artif Intell 1980;13:81–132.
19. Lewis D. Counterfactuals. Oxford, UK: Blackwell; 1973.
20. Kraus S, Lehmann D, Magidor M. Nonmonotonic reasoning, preferential models and cumulative logics. Artif Intell 1990;44:167–207.
21. Georgeff M, Lansky A. Reactive reasoning and planning. In: Proc Sixth National Conference on Artificial Intelligence (AAAI-87). Menlo Park, CA: AAAI Press; 1987. pp 677–682.
22. Rao A, Georgeff M. An abstract architecture for rational agents. In: Proc Int Conf on Knowledge Representation and Reasoning (KR'92). San Francisco: Morgan Kaufmann; 1992. pp 439–449.
23. Rao A, Georgeff M. BDI agents: From theory to practice. In: Proc First International Conference on Multi-Agent Systems (ICMAS'95). Menlo Park, CA: AAAI Press; 1995. pp 312–319.
24. Schild K. On the relationship between BDI logics and standard logics of concurrency. Autonom Agent Multi Agent Syst 2000;3:259–283.
25. van der Torre L, Tan Y. Contrary-to-duty reasoning with preference-based dyadic obligations. Ann Math Artif Intell 1999;27:49–78.
26. von Wright GH. Deontic logic: As I see it. In: McNamara P, Prakken H, editors. Norms, logics and information systems. New studies on deontic logic and computer science. Amsterdam: IOS Press; 1999. pp 15–25.
27. Alchourrón CE, Bulygin E. The expressive conception of norms. In: Hilpinen R, editor. New studies in deontic logic: Norms, actions and the foundations of ethics. Dordrecht, The Netherlands: D. Reidel; 1981. pp 95–124.

28. Alchourrón CE. Philosophical foundations of deontic logic and the logic of defeasible conditionals. In: Meyer J-J, Wieringa R, editors. Deontic logic in computer science: Normative system specification. New York: John Wiley; 1993. pp 43–84.
29. Makinson D. On a fundamental problem of deontic logic. In: McNamara P, Prakken H, editors. Norms, logics and information systems. New studies on deontic logic and computer science. Amsterdam: IOS Press; 1999. pp 29–54.
30. Veltman F. Defaults in update semantics. J Philos Logic 1996;25:221–261.
31. van Benthem J. Exploring logical dynamics. Stanford, CA: CSLI Publications; 1996.
32. Dignum F, Morley D, Sonenberg E, Cavedon L. Towards socially sophisticated BDI agents. In: Proc Fourth International Conference on MultiAgent Systems (ICMAS-2000). Menlo Park, CA: AAAI Press; 2000. pp 111–118.
33. von Wright GH. Deontic logic. Mind 1951;60:1–15.
34. Broersen JM, Wieringa RJ, Meyer J-JCh. A fixed-point characterization of a deontic logic of regular action. Fundamenta Informaticae 2001;49:107–128.
35. Wieringa RJ, Meyer J-JCh. Applications of deontic logic in computer science: A concise overview. In: Meyer J-JCh, Wieringa RJ, editors. Deontic logic in computer science: Normative system specification. New York: Wiley; 1993. pp 17–40.
36. Dastani M, van der Torre L. Programming boid agents: A deliberation language for conflicts between mental attitudes and plans. In: Proc Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS'04). New York: ACM Press; 2004. pp 706–713.
37. Engelfriet J, Jonker CM, Treur J. Compositional verification of multi-agent systems in temporal multi-epistemic logic. J Logic Lang Inform 2002;11:195–225.
38. Jonker CM, Treur J. Compositional verification of multi-agent systems: A formal analysis of pro-activeness and reactiveness. In: de Roever WP, Langmaack H, Pnueli A, editors. Proc International Workshop on Compositionality, COMPOS'97, Lecture Notes in Computer Science 1536. Berlin: Springer Verlag; 1998. pp 350–380.
39. de Roever WP, Langmaack H, Pnueli A, editors. Proc International Workshop on Compositionality, COMPOS'97. Lecture Notes in Computer Science 1536. Berlin: Springer Verlag; 1998.