Relating functionality descriptions to proof rules of input/output logic

Jan Broersen [a] Mehdi Dastani [b] Leon van der Torre [a]

[a] Vrije Universiteit Amsterdam
[b] Utrecht University

Abstract

We give a logical account of functionality desriptions of reasoning components such as defined in DESIRE. To this end we define a mapping of functionality descriptions to proof rules of input/output logics. This enables us to proof that properties such as 'monotony' and 'self boundedness' for functionality descriptions correspond to 'strengthening of the input' and 'transitivity' for input/output logic, respectively. The results are relevant for the logical study of the interactions of reasoning components, and for the development of specification and verification languages for the BOID architecture and similar architectures.

1 Introduction

The structured approach to the specification of agent systems, based on a notion of compositional architecture, has been called *compositional design* [1], and an example of a compositional design method is the DEsign and Specification of Interacting REasoning components (DESIRE) discussed in [1]. The reasoning process is structured according to a number of reasoning components that interact with each other. The functioning of the overall agent system is based on the functionality of the components plus the composition relation by which they are glued together. Specification of a composition relation may involve, for example, the possibilities for information exchange between components and the control that activates the components. The functionality of a reasoning component can be described by functionality descriptions, introduced in [12].

In this paper we study the problem of how to give a logic account of the reasoning behaviour of functionality descriptions in DESIRE. We do not develop a new logic, but show that functionality descriptions can be mapped to rules of input/output logic [9, 10]. This enables us to relate certain well-defined properties of functionality descriptions to specific inference rules of input output logics. The results are relevant for the logical study of the interactions of reasoning components [5], and for the development of specification and verification languages for the

BOID architecture (B,O,I, and D for Beliefs, Obligations, Intentions and Desires respectively [4]) and similar architectures.

The layout of this paper is as follows. In Section 2 we describe functionality descriptions in DESIRE. In section 3 we define Input-Output logic specifications of functionality descriptions. Section 4 mentions topics for future research, and the final section gives some concluding remarks.

2 Functionality descriptions ...

A functionality description is the set of input/output possibilities 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 as well makes use of information 'from the outside' as input. In this sense we can say that the component's 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 what facts can be derived in a given situation, and not at what time and in which order specific facts actually are derived.

Reasoning components are able to draw partial conclusions if only 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. Originally this notion was defined in terms of a three valued propositional partial logic. In this section we give a logical characterization of these declarative functionality descriptions by assuming that the input and output of a component are represented by propositional formulas instead of partial models. This may be accomplished through a standard reduction of three valued logic into propositional logic (which may not be efficient in either time or space). We also use the propositional derivation relation ($\vdash$) as the syntactic counterpart of the refinement relation and consider mappings which are defined on consistent propositions rather than on models.

Definition 1 *Let Σ be a finite set of atomic propositions. Then, a proposition F is called complete with respect to Σ if and only if it is consistent and for all atomic propositions $p \in \Sigma$ it holds either $F \vdash p$ or $F \vdash \neg p$.*

In the following, we split the set of atomic propositions Σ into Σ_{in} and Σ_{out} to indicate the subsets of atomic *input* and *output* propositions, respectively. Complete input and output propositions are then built on the basis of Σ_{in} and Σ_{out}, respectively. Now, we define various properties of a mapping that is defined on consistent input propositions relative to Σ.

Definition 2 *Let Σ be a finite signature, Σ_{in} its subset of atomic input propositions, α a mapping assigning propositions to input propositions, and F_{in} the set of all complete input propositions with respect to Σ_{in}, then:*

1. *The mapping α is called* conservative *if for all consistent input propositions F it holds: $\alpha(F) \vdash F$*

2. *The mapping α is called* monotonic *if for all consistent input propositions F, F' it holds: $F' \vdash F \Rightarrow \alpha(F') \vdash \alpha(F)$*

3. *The mapping α is called* self-bounded *if for all consistent input propositions F, F' it holds: $\alpha(F') \vdash F \Rightarrow \alpha(F') \vdash \alpha(F)$*

4. *The mapping α is called* well-informed *if for any literal p and for any consistent input proposition F, it holds: $(\forall F' \in F_{in}[F' \vdash F \Rightarrow \alpha(F') \vdash p]) \Rightarrow \alpha(F) \vdash p$*

3 ... and input/output logics

Input/output logic (IOL) [9, 10] takes a process oriented view on logic inference (like Reiter's default logic). Due to the focuss on the process of logic inference instead of on the traditional ('Tarskian') logical invariants, input/output logics are suitable for a variaty of reasoning tasks that are hard to formalize in traditional (modal) logics. In particular, IOLs do in general not assume that inputs for the reasoning process are included in the outputs, and inferences made in IOLs are not necessarily reversible (they cannot be used contrapositively). Note that *out* in input/output logics has a different meaning than *out* in the functionality descriptions, since it refers to output formulas of a component instead of to output atoms.

Definition 3 (Input/output logic) *[9][1] Let G be a set of pairs of propositional formulas $\{(a_1, x_1), \ldots, (a_n, x_n)\}$, read as 'if input a_1 then output x_1', etc., and consider the following proof rules: strengthening of the input (SI), conjunction for the output (AND), weakening of the output (WO), disjunction of the input (OR), cumulative transitivity (CT), transitivity (TRANS) and identity (Id) defined as follows:*

$$\frac{(a,x)}{(a \wedge b, x)}SI \qquad \frac{(a,x)(b,x)}{(a \vee b, x)}OR$$

$$\frac{(a,x)(a,y)}{(a, x \wedge y)}AND \qquad \frac{(a,x)}{(a, x \vee y)}WO$$

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

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

The output operation out_G is a mapping from propositions to propositions, defined indirectly by $x \in out_G(a) \Leftrightarrow (a,x) \in out(G)$. The following eight output operators have been defined as closure operators on the set G using the rules above.

[1]To facilitate the semantics of the output operations the generating set G is extended with $(\top, \top)$ such that it is never empty. Since we do not consider semantics in this paper we also do not consider this borderline case.

- out_1: *SI+AND+WO (simple-minded output)*
- out_2: *SI+AND+WO+OR (basic output)*
- out_3: *SI+AND+WO+CT (simple-minded reusable output)*
- out_4: *SI+AND+WO+OR+CT (basic reusable output)*
- out_i^+: out_i*+Id (throughput)*

Definition 4 below defines a relation between the mapping α of Definition 2 to the out_G operator of input/output logic of Definition 3. Before we relate the two, we have to make some simplifying assumptions about both mappings, which leads to Definition 4 below.

Single/multiple output. α maps the input to a single output, whereas *out* maps the input to a set of outputs. This problem is solved as follows. On the one hand, the single output F' of α implicitly represents the set $\{F \mid F' \vdash F\}$, as can be understood from the construction of a functionality description [11]. However, in the finite case the output of the operations studied in input/output logic can all be identified with a single formula due to WO and AND. In the semantics of input/output logic this means that all output relations are a deductively closed set.

Generators. out_G is defined relative to a set of rules or generators, whereas there are no rules defined in α. We therefore abstract from G and focus on the proof rules.

In- and output variables. In input/output logics there is no distinction between in- and output variables, and we therefore assume that all input variables are also output variables, and the set of intermediate variables is empty.

Definition 4 *Let α be a mapping as given in Definition 2. The interpretation mapping Π assigning input-output logic specifications out to mappings α is defined as follows:* $x \in \Pi(\alpha)(a) \Leftrightarrow \alpha(a) \vdash x$

In Definitions 2 and 3, we have introduced properties of functionality description mappings and input/output logics, respectively. We now show the correspondence between these properties for functional description mappings and their corresponding input/output logics.

Theorem 1 *Given α as in Definition 2, the following hold:*

1. *α is conservative iff $\Pi(\alpha)$ satisfies identity (Id)*
2. *α is monotonic iff $\Pi(\alpha)$ satisfies strengthening of the input (SI)*
3. *α is self-bounded iff $\Pi(\alpha)$ satisfies transitivity (TRANS)*
4. *if α is monotonic, then it is well-informed iff $\Pi(\alpha)$ satisfies OR*

Proof.

1. *α is conservative iff $\Pi(\alpha)$ satisfies Id*

   α is conservative $\Leftrightarrow$
   $\forall y\ \alpha(y) \vdash y \ \Leftrightarrow$
   $\forall y\ y \in \Pi(\alpha)(y)$.

2. *α is monotonic iff $\Pi(\alpha)$ satisfies SI*

   (monotonicity) $\Leftrightarrow\ a \wedge b \vdash a \Rightarrow \alpha(a \wedge b) \vdash \alpha(a) \Leftrightarrow$
   $\forall x\ [\alpha(a) \vdash x \Rightarrow \alpha(a \wedge b) \vdash x] \Leftrightarrow$
   $\forall x\ [x \in \Pi(\alpha)(a) \Rightarrow x \in \Pi(\alpha)(a \wedge b)]$

3. *α is self-bounded iff $\Pi(\alpha)$ satisfies $TRANS$*

   $\forall x, y\ [\alpha(x) \vdash y \ \rightarrow\ \alpha(x) \vdash \alpha(y)] \Leftrightarrow$
   $\forall x, y, z\ [\alpha(x) \vdash y \ \rightarrow\ (\alpha(y) \vdash z \rightarrow \alpha(x) \vdash z)] \Leftrightarrow$
   $\forall x, y, z\ [\alpha(x) \vdash y \ \wedge\ \alpha(y) \vdash z \rightarrow \alpha(x) \vdash z] \Leftrightarrow$
   $\forall x, y, z\ [y \in \Pi(\alpha)(x) \ \wedge\ z \in \Pi(\alpha)(y) \rightarrow z \in \Pi(\alpha)(x)]$

4. *if α is monotonic, then it is well-informed iff $\Pi(\alpha)$ satisfies OR*

   ("$\Rightarrow$")
   We assume the well-informedness and the antecedent of OR, and prove the consequent of OR. First, consider the set $\mathcal{F} = \{F' \mid F' \vdash a \vee b\}$ of complete input propositions with respect to Σ_{in}. Since each F' is a complete input proposition, it follows that for all F' it holds $F' \vdash a$ or $F' \vdash b$. By monotonicity, we obtain that for all F' it holds $\alpha(F') \vdash \alpha(a)$ or $\alpha(F') \vdash \alpha(b)$ (*). Now, consider the antecedent of the OR and assume $\alpha(a) \vdash x$ and $\alpha(b) \vdash x$ for some arbitrary literal x. Then, by transitivity of the propositional derivation relation, these assumptions together with (*) imply $\alpha(F') \vdash x$. Using the antecedent of OR, we can thus conclude that for any proposition x with $\alpha(a) \vdash x$ and $\alpha(b) \vdash x$ it holds $(\forall F'[F' \vdash a \vee b \Rightarrow \alpha(F') \vdash x])$. Applying this conclusion to the well-informedness, we can then conclude $\alpha(a \vee b) \vdash x$ which is $x \in \Pi(\alpha)(a \vee b)$.

   ("$\Leftarrow$")
   We use the generalized form of OR, assume that for any arbitrary x it holds $\forall F'[F' \vdash F \Rightarrow \alpha(F') \vdash x]$, and prove $\alpha(F) \vdash x$. Consider again the set $\mathcal{F} = \{F' \mid F' \vdash F\}$ of complete input propositions with respect to Σ_{in}. Given any arbitrary proposition x and for all $F'_i \in \mathcal{F}$, we write the general form of OR as follows: $\bigwedge_i(\alpha(F'_i) \vdash x) \Rightarrow \alpha(\bigvee_i F'_i) \vdash x$ for all $i = 1, \ldots, n$ and any $n > 1$. Assume that for arbitrary x it holds $\forall F'[F' \vdash F \Rightarrow \alpha(F') \vdash x]$ (*).

For all $i = 1, \ldots, n$ and any $n > 1$, F must then be of the form $\bigvee_i F'_i$ where $F'_i \in \mathcal{F}$. But, then the assumption (*) states $\forall F' \in \mathcal{F} : \alpha(F') \vdash x$ and thus we have $\bigwedge_i (\alpha(F'_i) \vdash x)$. Applying the generalized OR to this result, we obtain $\alpha(\bigvee_i F'_i) \vdash x$. We have argued that F is of the form $\bigvee_i F'_i$ which implies $\alpha(F) \vdash x$. (note that for this direction we did not need monotonicity).

The following example illustrates Theorem 1.

Example 1 *Consider the following mapping α which is not a functionality description [11, Fig. 13]: $\alpha(\neg p_1) = \neg p_1$; $\alpha(p_1) = p_1$; $\alpha(p_1 \vee \neg p_1) = p2$. Its interpretation mapping $\Pi(\alpha)$ is then such that $(\neg p_1, p_2), (p_1, p_2) \notin \Pi(\alpha)$ whereas $(\top, p_2) \in \Pi(\alpha)$. It thus violates SI.*

Consider the following not well-informed functionality description α [11, Fig. 15]: $\alpha(\neg p_1) = \neg p_1 \wedge p_2$; $\alpha(p_1) = p_1 \wedge p_2$. Its interpretation mapping $\Pi(\alpha)$ is then such that $(\neg p_1, p_2), (p_1, p_2) \in \Pi(\alpha)$ whereas $(\top, p_2) \notin \Pi(\alpha)$. It thus violates OR.

Summarizing, Theorem 1 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. The major issue for further research for the foundations of compositional architectures is the logical analysis of the *interaction* of input/output components [11, 7, 2, 12], such as the distinction between object and meta level interactions (for which hierarchical logics may be used) and feedback loops between components (for which generalized self-boundedness or generalized CT rules may be used).

4 Further research

There are several consequences of Theorem 1 for further logical investigations.

First, the declarative functionality descriptions can be compared to other logical relations. 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, i.e. is equivalent to out_2^+) [9]. 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.

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

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 [8] and more recently by

[6]: Id+CT is cumulative inference, Id+CT+OR is preferential inference. Another possible extension is to allow for components in which the input is not included in the output. Input/output logic focuses on systems without Id.

Third, with the partition 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, i.e. 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, i.e. we can use out_1 and out_2. This shows the use of all these different systems: The more complex combination of variables is allowed, the more extended proof system (input/output logic) is required. Note 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^+).

Fourth, another issue of relevance for complex compositional systems is their computational complexity. 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).

5 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 operators. 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 between components, and control.

The first step for a logical investigation into the interactions of components is a logical description of the components themselves. For the specification of individual components we have related functionalities of reasoning components to input/output logic [9, 10], in the sense that it has been shown how the formalization of the functionality description of a reasoning component, introduced in [11], 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 between 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.

References

[1] F.M.T. Brazier, C.M. Jonker, and J. Treur. Principles of compositional multi-agent system development. In J. Cuena, editor, *Proceedings of the 15th IFIP*

World Computer Congress, WCC'98, Conference on Information Technology and Knowledge Systems, IT&KNOWS'98, pages 347–360, 1998. To be published by IOS Press.

[2] F.M.T. Brazier and J. Treur. Managing conflicts in reflective agents. In H.-J. Mueller and R. Dieng, editors, *Computational Conflicts: Conflict Modeling for Distributed Systems*, pages 63–81. Springer Verlag, 2000.

[3] G. Brewka. Reasoning about priorities in default logic. In *Proceedings of the AAAI-94*, 1994.

[4] J.M. Broersen, M. Dastani, J. Hulstijn, and L.W.N. van der Torre. Goal generation in the BOID architecture. *Cognitive Science Quarterly Journal*, 2002. forthcomming.

[5] J.M Broersen, M. Dastani, and L.W.N. van der Torre. Resolving conflicts between beliefs, obligations, intentions and desires. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty. Proceedings of ECSQARU'01.* Springer, 2001.

[6] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.

[7] P. Leemans, J. Treur, and M. Willems. On the verification of knowledge-based reasoning modules. Technical Report IR-346, Vrije universiteit, faculteit der wiskunde en informatica, 1993.

[8] D. Lewis. *Counterfactuals.* Blackwell, Oxford, 1973.

[9] D. Makinson and L.W.N. van der Torre. Input-output logics. *Journal of Philosophical Logic*, 29:383–408, 2000.

[10] D. Makinson and L.W.N. van der Torre. Constraints for input-output logics. *Journal of Philosophical Logic*, 30:155–185, 2001.

[11] J. Treur. Declarative functionality descriptions of interactive reasoning modules. In H. Boley and M.M. Richter, editors, *Processing Declarative Knowledge, Proc. of the International Workshop PDK-91*, volume 567 of *Lecture Notes in AI*, pages 221–236. Springer, 1991. Also extended work report.

[12] J. Treur. Semantic formalisation of interactive reasoning functionality. *International Journal of Intelligent Systems*, To appear.