Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 141 (2005) 135–162
www.elsevier.com/locate/entcs

Interaction in Normative Multi-Agent Systems

Guido Boella[1]

Dipartimento di Informatica
Università di Torino
Torino, Italy

Joris Hulstijn[2]

Faculty of Economics and Business Administration
Vrije Universiteit
Amsterdam, The Netherlands

Leendert van der Torre[3]

CWI Amsterdam
and Delft University of Technology
The Netherlands

Abstract

The central research question of this paper is how notions developed in interactive computing such as abstract behavior types, the coordination language Reo, and Boolean circuits with registers, can be used to extend logical input/output nets, or lions for short. Lions are based on input/output logic, a deontic logic which is not used as a (non-classical) inference engine deriving output from input, but as a secretarial assistant for logically assisted transformations from input to output. We consider two extensions of input/output logics and lions. First, we consider input/output logics defined on infinite sequences (or streams) of inputs and outputs. Secondly, we consider lions with AND and register gates, formalizing the behavior of channels and connectors. We discuss also the role of interactive computing in normative multi-agent systems motivating the development of lions.

Keywords: Normative systems, multi-agent systems, deontic logic, input/output logic, coordination, communication.

doi:10.1016/j.entcs.2005.05.020

1 Introduction

According to many in computer science, the interaction paradigm provides a new conceptualization of computational phenomena that emphasize interaction rather than algorithms: concurrent, distributed, reactive, embedded, component-oriented, agent-oriented and service-oriented systems all exploit interaction as a fundamental paradigm [34,5].

In this paper we consider Makinson and van der Torre's logical input/output nets [25], or lions for short, as a model for interactive computation. Lions are structured assemblies of input/output operations and extend Makinson and van der Torre's input/output logics [22,23]. They are graphs, with the nodes labelled by pairs (G, out) where G is a normative code and out is an input/output operation. The edges of the graph represent channels, which indicate which nodes have access to which other nodes and provide passage for the transmission of local outputs as local inputs. The graph is further equipped with an entry point and an exit point, for global input and output.

We consider also two extensions of lions in this paper, both inspired by the work of Arbab and colleagues on abstract behavior types [4], the coordination language Reo [3], and Boolean circuits with registers [28].

Streams. Instead of considering one input at a time, we consider input/output logics on infinite sequences (or streams) of inputs and outputs.

Connectors. We consider AND and register gates to compose channels into connectors (or circuits).

Finally we discuss the role of interaction in normative multi-agent systems, and how it is used to motivate the further development of lions. Our investigations reveal a huge number of possible further extensions of lions, which raises the question which extensions should be studied next. Input/output logic originates from deontic logic, a branch of philosophical logic that studies logical relations among obligations, permissions and prohibitions, and which has been used to model legal and moral systems, as well as problems in computer science that involve constraints that can be violated [35]. Whereas deontic logic has been very helpful in the development of input/output logics, it does not seem very helpful to guide the development of lions. A motivation of lions comes from agent architectures and normative multi-agent systems, i.e., "sets of agents (human or artificial) whose interactions can fruitfully be regarded as norm-governed; the norms prescribe how the agents ideally should and should

[1] Email: guido@di.unito.it
[2] Email: jhulstijn@feweb.vu.nl
[3] Email: torre@cwi.nl

not behave" [21].

The layout of this paper is as follows. In Section 2 we repeat the definitions of input/output logic, and in Section 3 we explain how the concept of 'logic as a secretarial assistant' is related to interactive computing. In Section 4 we discuss input/output logics of streams and register and AND gates for channels and connectors. In Section 5 we discuss interactive computing in normative multi-agent systems.

2 Makinson and van der Torre's input/output logic

Makinson and van der Torre [22] argue that the new role of logic is not to study some kind of non-classical logic, but a way of using the classical one. From a very general perspective, logic is often seen as an 'inference engine', with premises as inputs and conclusions as outputs. Instead it may be seen in another role: as a 'secretarial assistant' to some other, perhaps non-logical, transformation engine. From this point of view, the task of logic is one of preparing inputs before they go into the machine, unpacking outputs as they emerge and, less obviously, *coordinating* the two. The process is one of 'logically assisted transformation', and is an inference only when the central transformation is so. The general perspective is one of 'logic at work' rather than 'logic in isolation'. The brief description in this section is taken from Makinson and van der Torre's introduction to input/output logic [25]. Please refer to the original papers for further explanations and motivations [22,23].

2.1 Unconstrained Input/Output Operations

Imagine a black box into which we may feed propositions as input, and that also produces propositions as output. Of course, classical consequence may itself be seen in this way, but it is a very special case, with additional features: inputs are themselves outputs, since any proposition classically implies itself, and the operation is in a certain sense reversible, since contraposition is valid. However, there are many examples of logical transformations without those features. In [22], the outputs either express some kind of belief or expectation, or some kind of desirable situation in the conditions given by the inputs.

Technically, a normative code is seen as a set G of conditional norms, i.e. a set of such ordered pairs (a, x). For each such pair, the body a is thought of as an input, representing some condition or situation, and the head x is thought of as an output, representing what the norm tells us to be desirable, obligatory or whatever in that situation. The task of logic is seen as a modest one. It is not to create or determine a distinguished set of norms, but rather to prepare information before it goes in as input to such a set G, to unpack

output as it emerges and, if needed, coordinate the two in certain ways. A set G of conditional norms is thus seen as a transformation device, and the task of logic is to act as its 'secretarial assistant'.

In the simplest kind of unconstrained input/output operations, a set A of propositions serves as explicit input, which is prepared by being expanded to its classical closure $Cn(A)$. This is then passed into the 'black box' or 'transformer' G, which delivers the corresponding immediate output $G(Cn(A)) = \{x : \exists a \in Cn(A), (a, x) \in G\}$. Finally, this is expanded by classical closure again into the full output $out_1(G, A) = Cn(G(Cn(A)))$. We call this simple-minded output. This is already an interesting operation. As desired, it does not satisfy the principle of identity, which in this context we call throughput, i.e. in general we do not have $a \in out_1(G, \{a\})$, which we write briefly, dropping the parentheses, as $out_1(G, a)$.

The input/output operation out_1 is characterized by three rules. Writing $x \in out_1(G, a)$ as $(a, x) \in out_1(G)$ and dropping the right hand side as G is held constant, these rules are:

- Strengthening Input (SI): From (a, x) to (b, x) whenever $a \in Cn(b)$
- Conjoining Output (AND): From (a, x), (a, y) to $(a, x \wedge y)$
- Weakening Output (WO): From (a, x) to (a, y) whenever $y \in Cn(x)$.

However, simple-minded output lacks features that may be desirable in some contexts. In the first place, the preparation of inputs is not very sophisticated. Consider two inputs a and b. If $x \in Cn(a)$ and $x \in Cn(b)$ then $x \in Cn(a \vee b)$, but if $x \in out_1(G, a) = Cn(G(Cn(a)))$ and $x \in out_1(G, b) = Cn(G(Cn(b)))$ then we cannot conclude $x \in out_1(G, a \vee b) = Cn(G(Cn(a \vee b)))$.

In the second place, even when we do not want inputs to be automatically carried through as outputs, we may still want outputs to be reusable as inputs – which is quite a different matter. Operations satisfying each of these two features can be provided with explicit definitions, characterized by straightforward rules. We thus have four very natural systems of input/output, which are labelled as follows: simple-minded alias out_1 (as above), basic (simple-minded plus input disjunction: out_2), reusable (simple-minded plus reusability: out_3), and reusable basic (all together: out_4).

The three stronger systems may also be characterized by adding one or both of the following rules to those for simple-minded output:

- Disjoining input (OR): From (a, x), (b, x) to $(a \vee b, x)$
- Cumulative transitivity (CT): From (a, x), $(a \wedge x, y)$ to (a, y).

These four operations have four counterparts that also allow throughput. Intuitively, this amounts to requiring $A \subseteq G(A)$. In terms of the definitions, it

is to require that G is expanded to contain the diagonal, i.e. all pairs (a, a). Derivationally, it is to allow arbitrary pairs of the form (a, a) to appear as leaves of a derivation; this is called the zero-premise identity rule ID. All eight systems are distinct, with one exception: basic throughput, which we write as out_2^+, authorizes reusability, so that $out_2^+ = out_4^+$. This is shown by the derivation in Figure 1, which also serves here to illustrate the proof theory of input/output logic. In the final step (OR) we are also implicitly using a replacement of classically equivalent propositions (or an application of SI).

$$\dfrac{\dfrac{\dfrac{\dfrac{(a,x)}{(a\wedge\neg x,x)}\,\text{SI}\quad\dfrac{-}{(a\wedge\neg x,a\wedge\neg x)}\,\text{ID}}{(a\wedge\neg x,x\wedge(a\wedge\neg x))}\,\text{AND}}{(a\wedge\neg x,y)}\,\text{WO}\qquad (a\wedge x,y)}{(a,y)}\,\text{OR}$$

Fig. 1. Basic throughput authorizes reusability

2.2 Constraints

The motivation of constraints comes from the logic of conditional norms, where we need an approach that does not presume that directives carry truth-values. Unconstrained input/output provides us with a simple and elegant construction, with straightforward behavior, but whose application to norms totally ignores the subtleties of violations and exceptions. Therefore input/output operations may be subjected to consistency constraints [23]. Our strategy is to adapt a technique that is well known in the logic of belief change - cut back the set of norms to just below the threshold of making the current situation contrary-to-duty. In effect, we carry out a contraction on the set G of norms.

Specifically, we look at the maximal subsets $G' \subseteq G$ such that $out(G', A)$ is consistent with input A. In [23], the family of such G' is called the maxfamily of (G, A), and the family of outputs $out(G', A)$ for G' in the maxfamily, is called the outfamily of (G, A). To illustrate this, consider the set of norms $G = \{(\top, \neg(f \vee d)), (d, f \wedge w)\}$, where $\top$ stands for any tautology, with the contrary-to-duty input d. Using simple-minded output, maxfamily(G, d) has just one element $\{(d, f \wedge w)\}$, and so outfamily(G, d) has one element, namely $Cn(f \wedge w)$.

Although the outfamily strategy is designed to deal with contrary-to-duty norms, its application turns out to be closely related to belief revision and non-monotonic reasoning when the underlying input/output operation authorizes throughput. When all elements of G are of the form $(\top, x)$, then for the degenerate input/output operation $out_2^+(G, a) = out_4^+(G, a)$, the elements of

outfamily(G, a) are just maxichoice revisions, in the sense of belief revision as defined by Alchourrón, Gärdenfors and Makinson [1]. These coincide, in turn, with the extensions of the default system of Poole [26]. More surprisingly, there are close connections between out_3^+ and the default logic of Reiter [27], see [23] for a further discussion on constraints in input/output logics and their relation to constraints in belief revision and non-monotonic reasoning.

2.3 Permissions

Moreover, input/output logics also provide a convenient platform for distinguishing and analyzing several different kinds of permission [24]. They give a clear formal articulation of the well-known distinction between negative and positive permission. In philosophical discussion of norms it is common to distinguish between two kinds of permission, negative and positive. Negative permission is easy to describe: something is permitted by a code iff it is not prohibited by that code, i.e. iff nihil obstat. In other words, taking prohibition in the usual way, something is negatively permitted by a code iff there is no obligation to the contrary. From the point of view of input/output logic, negative permission is straightforward to define: we simply put $(a, x) \in$ negperm(G) iff $(a, \neg x) \notin out(G)$, where *out* is any one of the four input/output operations that we have already discussed.

Positive permission is more elusive. As a first approximation, one may say that something is positively permitted by a code if and only if the code explicitly presents it as such. Makinson and van der Torre distinguish what they call forward and backward permission as two distinct kinds of positive permission. Forward permission answers to the needs of the citizen, who needs to know whether an action that he is entertaining is permitted in the current situation. It also corresponds to the needs of authorities assessing the action once it is performed. If there is some explicit permission that covers the action in question, then it is itself implicitly permitted. On the other hand, backward permission fits the needs of the legislator, who needs to anticipate the effect of adding a prohibition to an existing corpus of norms. If prohibiting x in condition a would commit us to forbid something that is implicit in what has been expressly permitted, then adding the prohibition is inadmissible under pain of incoherence, and the pair (a, x) is to that extent protected from prohibition.

Intuitively, forperm tells us that (a, x) is permitted whenever there is some explicitly given permission (c, z) such that when we treat it as if it were an obligation, joining it with G and applying the output operation to the union, then we get (a, x). Permissions are thus treated like weak obligations, the only difference being that while the latter may be used jointly, the former may only

be applied one by one. Backperm tells us that (a, x) is permitted whenever, given the obligations already present in G, we can't forbid x under the condition a without thereby committing ourselves to forbid, under a condition c that could possibly be fulfilled, something z that is implicit in what has been expressly permitted.

The proof theory of the various kinds of permissions contains various unexpected properties and proof-theoretic novelties. For example, whenever *out* satisfies a Horn rule, then the corresponding negperm operation satisfies an inverse one. Forperm and backperm are very different operations. Whereas forperm satisfies SI, backperm satisfies weakening of the input WI. Like negative permission, backperm satisfies the inverse rule of any Horn rule satisfied by out; but forperm satisfies instead a subverse rule. See [24] for the details.

Permissions in the context of constraints have not been considered yet.

2.4 Lions

Structured assemblies of input/output operations, called logical input/output nets, or lions for short, are graphs, with the nodes labeled by pairs (G, out) where G is a normative code and out is an input/output operation (or recursively, by other lions). The relation of the graph indicates which nodes have access to others, providing passage for the transmission of local outputs as local inputs. The graph is further equipped with an entry point and an exit point, for global input and output. Lions have not been extended with constraints or permissions yet.

3 Lions for exogenous coordination

Despite its origin in deontic logic, input/output logic is clearly related to functionality descriptions of components in interactive computing. For example, it has been shown [15] how the proof rules of input/output logic, and thus input/output operations, are related to Treur's functionality descriptions of interactive reasoning components [33]. Also constraints and lions can be interpreted in the context of interactive computing. At this moment it is not clear to us whether there is an analogue to permissions in interactive computing.

Moreover, the secretarial role of logic formalized by input/output logic can be related to discussions in interactive computing. For example, Arbab [2] argues that "coordination models and languages can also be classified as either endogenous or exogenous. [...] Exogenous models and languages provide primitives that support coordination of entities from without. In applications that use exogenous models primitives that affect the coordination of each module are outside the module itself." Typical examples of exogenous coordination

are protocols enforced by the environment.

In this section we consider the proof rules of input/output logics as properties that can be enforced on the components by exogenous coordination. In particular, it has been suggested by Makinson and van der Torre that the identity rule corresponds to a forward loop, and that the cumulative transitivity rule corresponds to a feedback loop. We use lions to make this idea more precise. The problem can be phrased as follows. How far can the various ways of strengthening the input/output operation out_1 to out_n (n = 2,3,4) with *out* without +, be simulated by integrating other familiar devices into the system as a whole? Before we introduce the definitions, we consider two examples.

3.1 The identity rule

Consider the lion in Figure 2. This figure should be read as follows. There are only three nodes, the start and end nodes without input/output logics associated with them, and a single node with which a component and therefore an input/output logic is associated. Moreover, there is an edge from the start node to the end node.

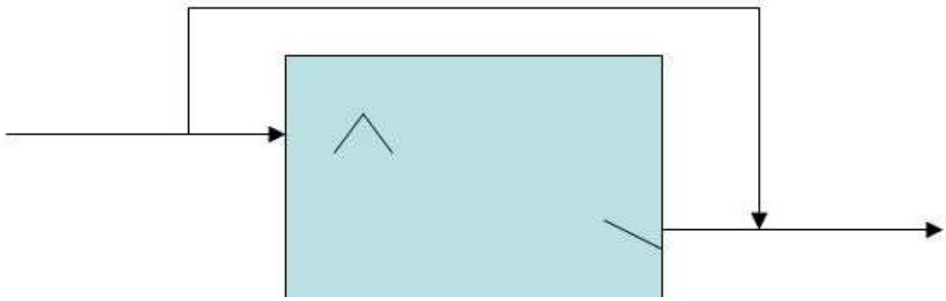

Fig. 2. Single component with feed forward (ID)

Informally, the behavior of the lion can be defined as follows, using some of Arbab's terminology.

- An edge between two nodes is called a channel from source to sink. The behavior of a channel is that the input of the sink contains (at least) the output of the source.
- When there are several channels ending in a node, then the input of this node is the union (or logical closure) of all the inputs. So at the right hand side of the black box in Figure 2, the output of the channel and the output of the component are combined.
- When there are several channels leaving a node, then the node's output is

replicated to all channels.

For example, assume that the box in Figure 2 is described by a simple-minded operation $out_1(G, A)$, then the output of the lion is $Cn(out_1(G, A) \cup A)$, which happens to be $out_1^+(G, A)$. So the component is exogenously (i.e., without being aware of it or able to influence it) coordinated so that the system as a whole behaves like a throughput operation, regardless of whether the component itself behaves as an input/output operation or a throughput operation.

3.2 Cumulative transitivity

As a second example, we consider the more complex case of a feedback loop in Figure 3, which leads to cumulative transitivity of the associated input/output logic. Again there are three nodes in the lion, but this time there is a feedback loop of the intermediate (or exit) node to the start (or intermediate) node.

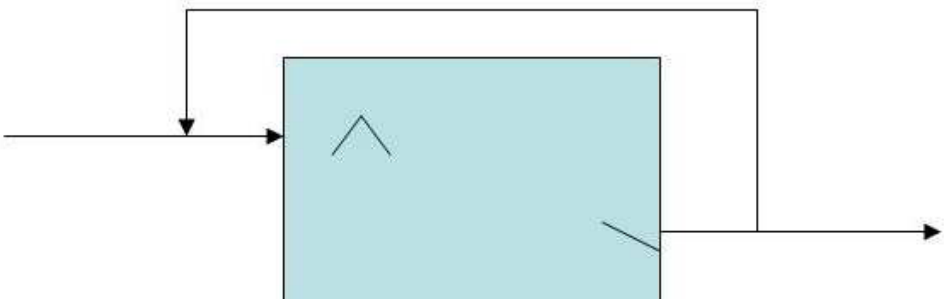

Fig. 3. Single component with feedback (CT)

The lion contains a cycle, and its behavior is therefore more complicated than the behavior of the lion in Figure 2. As usual in networks with feedback loops, we describe the behavior of the lion using a fixed point definition. The output of the lion is the least set of formulas such that the input of the black box contains at least its output. This is formalized in Definition 3.2 below.

For example, assume that the black box in Figure 3 can be described by a simple-minded operation $out_1(G, A)$, and therefore the output of the lion is $\cap\{Cn(G(B)) \mid A \subseteq B = Cn(B) \supseteq G(B)\}$, which happens to be $out_3(G, A)$. So the component is exogenously coordinated such that the system behaves like a reusable output operation, regardless of the fact whether the component itself behaves as a simple-minded or reusable operation.

To consider more complicated examples, we define lions formally based on the above intuitions. For simplicity we do not define recursive lions. Note that we do not require that all nodes of the lion are reachable from the start node, though this seems a reasonable extension.

Definition 3.1 Let L be a propositional logic. A logical input/output net or lion is a tuple $\langle N, s, e, E, G, O\rangle$ where N is a set of nodes, $s, e \in N$ the start and end node of the lion, $E \subseteq N \times (N \setminus \{s\})$ a set of edges, $G : (N \setminus \{s, e\}) \to 2^{L \times L}$ is a complete function from the set of nodes to sets of norms defined over L, and $O : (N \setminus \{s, e\}) \to \{out_1, \ldots, out_4^+\}$ a complete function associating an input/output operation to nodes.

The behavior of a lion is defined as a least fixed point, which due to monotonicity (and the Knaster-Tarski theorem) exists and is unique.

Definition 3.2 Let L be a propositional logic, A a set of formulas of L, and $\langle N, s, e, E, G, O\rangle$ a lion. Moreover, let $in(n) = \{m \mid (m, n) \in E\}$ be the set of nodes connected to n. A possible behavior of the lion is a function $b : N \to 2^L$ from the nodes to sets of propositional formulas such that:

(i) $b(s) = Cn(A)$: A is the behavior of the start node, and

(ii) $b(n) = O(n)(G(n), b(in(n)))$ for $n \in N \setminus \{s, e\}$, where we write $b(M) = \cup_{m \in M} b(m)$ for $M \subseteq N$: the behavior of an internal node is defined by the associated input/output logic, and

(iii) $b(e) = Cn(b(in(e)))$: the behavior of the end node is the union of the outputs of the nodes connected to the end node.

Moreover, consider the case in which the black box in Figure 2 is not simple-minded output, but reusable output $out_3(G, A)$. The behavior of the lion is $Cn(out_3(G, A) \cup A)$, which happens again to be equivalent to $out_3^+(G, A)$ [23]. Likewise, we consider the case in which the black box in Figure 3 is simple-minded throughput $out_1^+(G, A)$. The output of the lion is again $out_3^+(G, A)$.

3.3 The disjunction rule

Whereas the identity rule and the cumulative transitivity rule are naturally modeled as feed forward and feedback loops, this is not the case for the disjunction rule. In the semantics of input/output logics, this rule corresponds to reasoning by cases. We can consider reasoning by cases as a kind of exogenous coordination, in the sense that we can ensure that a component behaves like it is reasoning by cases, when it is not really doing so. The idea is that in the wrapper around the component, we need to generate the cases, and then we collect the outputs again.

We cannot model this using lions as defined above. In general, there seem to be two ways in which this can modeled using a network, since it can be modeled using a set of copies of the component running in parallel, or using a single component dealing with the cases sequentially. For example, consider

the network in Figure 4 in which the logical language contains three propositional atoms p, q, r and the input of the component is p. Then there are four cases which should be considered, $q \wedge r$, $q \wedge \neg r$, $\neg q \wedge r$, and $\neg q \wedge \neg r$.

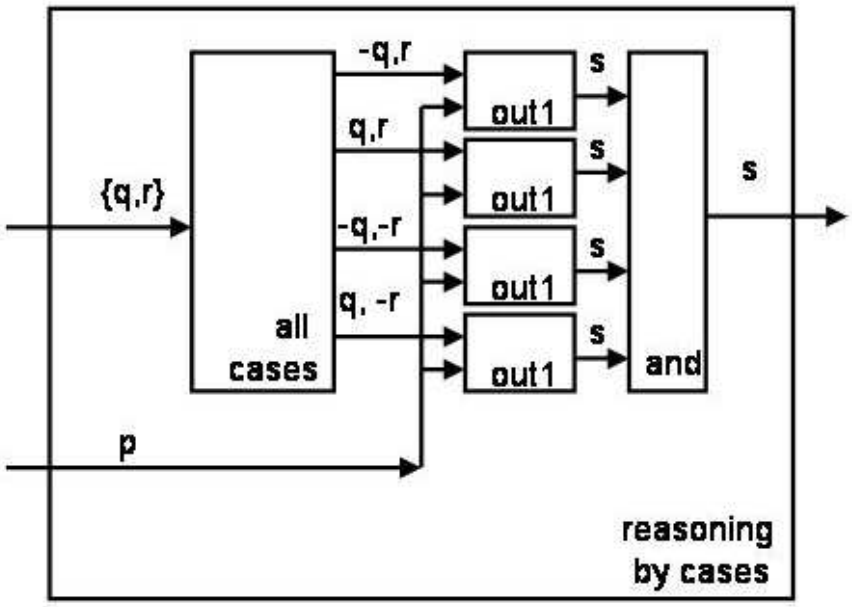

Fig. 4. Reasoning by cases as exogenous coordination

However, a drawback of this approach is that the number of cases is unknown and can be high or even infinite. The alternative of doing the cases in sequence uses a single component, by generating the cases one after the other, sending them through the component, and collecting them afterwards. This deals with the unknown number of cases, but not with an infinite number. Moreover, after each case we have to reset the component.

There are more problems related to reasoning by cases in the context of lions. For example, if the black box in Figure 2 is not simple-minded output, but basic output $out_2(G, A)$, then the behavior of the lion is weaker than $out_2^+(G, A)$. For example, $out_2^+(\{(a, x)\}, \emptyset) = Cn(a \rightarrow x)$, whereas the corresponding lion only derives the propositional tautologies. Roughly, the reason is that we only have feed forward for the whole lion, whereas in basic throughput we have feed forward for each case. For example, in Figure 4 we need feed forward from the node 'all cases' to the AND gate.

Due to these problems, one may be tempted to ignore the disjunction rule. However, we would like to emphasize the importance of this reasoning pattern in many kinds of reasoning. For example, a variant of reasoning by cases known as the sure thing principle is a key principle in Savage's classical

decision theory in the foundations of statistics [29].

3.4 Combining components

As an example of a lion with multiple components, consider the simple sequence of two components in the BD architecture in Figure 5. This lion represents the goal generation component of an agent architecture with a component outputting beliefs (B) and a component outputting desires (D), also see Section 5.1 for this interpretation.

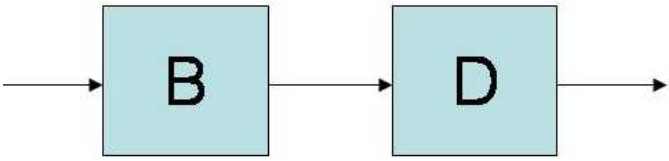

Fig. 5. BD

If B and D are input/output operations, then also the lion is an input/output operation. As David Makinson observed (personal communication), we can say quite generally the following. Take any lion $L = \langle N, s, e, E, G, O\rangle$. Define G' to be the set of all ordered pairs (a, x) such that x is in the output of L (i.e. $x \in b(e)$) when L is given input a (i.e. $a \in b(s)$), Then $G' = out_1(G')$. Reason: so defined, G' is closed under SI, AND, WO.

Moreover, the properties of the lion depend on the properties of the components. For example, when both B and D satisfy identity, then the lion satisfies the identity rule. Likewise, when both B and D satisfy cumulative transitivity, then the lion satisfies this rule. However, when B satisfies the disjunction rule, then the lion does not have to satisfy the disjunction rule.

Moreover, consider the BD architecture with a feedback loop over both components in Figure 6. Using the semantic Definition 3.2 we can show that this architecture is different from the architecture in Figure 5 when the components do not satisfy cumulative transitivity, and they are not equivalent either when the components satisfy cumulative transitivity.

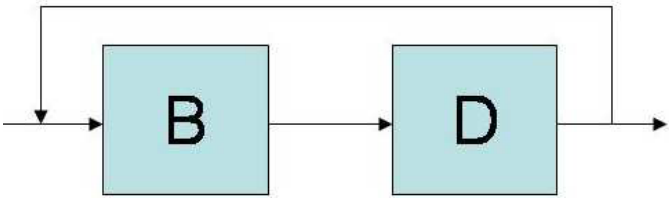

Fig. 6. BD with feedback

The proof theory of lions has not been studied yet, and neither the equivalence of lions.

4 Extensions to lions inspired by interactive computing

There are many ways in which lions can be extended to cover a wider range of interactive computing. In this section we consider two of those extensions based on input/output logics for streams, and connectors.

4.1 Input/output logic for streams

In his section we are inspired by the notion of abstract behavior type (ABT), which has been proposed by Arbab [4] as a proper foundation model of coordination - more precisely, as a higher-level alternative to abstract data type (ADT). The ABT model supports a much looser coupling than is possible with the ADT's operational interface. The point is that ABTs do not specify any detail about the operations that may be used to implement such behavior or the data types the system may manipulate for its realization - just like input/output logic does not specify the black box. ABTs are modelled as input/output operations on infinite sequences of data, called streams, which suggests that input/output logics should not only consider transformations at one moment in time, but also transformations over time. In this section we consider this extension.

4.1.1 Changing the base logic

The first input/output logic for streams we consider replaces the propositional base logic by a logic of streams, and then applies input/output logic on this logic of streams. We write α for an infinite sequence (called a stream) of propositions $\langle p_0, p_1, p_2, \ldots\rangle$, and we define Boolean connectives as point-wise applications of the connectives. Thus $\neg\alpha = \neg\langle p_0, p_1, p_2, \ldots\rangle = \langle\neg p_0, \neg p_1, \neg p_2, \ldots\rangle$. Moreover, for binary connectives, we have $\langle p_0, p_1, p_2, \ldots\rangle \wedge \langle q_0, q_1, q_2, \ldots\rangle = \langle p_0 \wedge q_0, p_1 \wedge q_1, p_2 \wedge q_2, \ldots\rangle$, etc. Likewise, $\langle p_0, p_1, p_2, \ldots\rangle \in Cn(\langle q_0, q_1, q_2, \ldots\rangle)$ if and only if $p_i \in Cn(q_i)$ for $i = 0, 1, 2, \ldots$.

Definition 4.1 Let L be a propositional language, let α, β, γ be infinite sequences (called streams) of elements of L, and let boolean connectives on such streams be defined as point-wise application of the connectives on elements of the streams.

Moreover, let the norms in G be pairs $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\}$, read as 'if input α_1, then output β_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), and cumulative transitivity (CT) and Identity (Id) defined as follows:

$$\frac{(\alpha,\gamma)}{(\alpha\wedge\beta,\gamma)}SI \qquad \frac{(\alpha,\beta),(\alpha,\gamma)}{(\alpha,\beta\wedge\gamma)}AND \qquad \frac{(\alpha,\beta\wedge\gamma)}{(\alpha,\beta)}WO$$

$$\frac{(\alpha,\gamma),(\beta,\gamma)}{(\alpha\vee\beta,\gamma)}OR \qquad \frac{(\alpha,\beta),(\alpha\wedge\beta,\gamma)}{(\alpha,\gamma)}CT \qquad \frac{}{(\alpha,\alpha)}Id$$

The following four output operators are defined as closure operators on the set G using the rules above.

out_1: SI+AND+WO	(simple-minded output)
out_2: SI+AND+WO+OR	(basic output)
out_3: SI+AND+WO+CT	(reusable output)
out_4: SI+AND+WO+OR+CT	(basic reusable output)

Moreover, the following four throughput operators are defined as closure operators on the set G.

out_i^+: out_i+Id (throughput)

We write $out(G)$ for any of these output operations, and $out^+(G)$ for any of these throughput operations, we write $G \vdash_{iol} (\alpha,\beta)$ for $(\alpha,\beta) \in out(G)$, or, depending on context, $(\alpha,\beta) \in out^+(G)$, and we write $G \vdash_{iol} G'$ iff $G \vdash_{iol} (\alpha,\beta)$ for all $(\alpha,\beta) \in G'$.

Makinson and van der Torre study only the given eight input/output operations, but this set of input/output logics is not thought to be exhaustive. For example, we believe that also operations not satisfying strengthening of the input may be of interest, and the extension of simple-minded output with the transitivity rule (derive (a,y) from (a,x) and (x,y)) may have some applications. In throughput systems, assuming the simple-minded rules, the transitivity rule is equivalent to cumulative transitivity, but this is not the case in general. Moreover, the replacement of propositional logic by a logic of streams may again lead to the study of new inference rules. However, we do not consider such alternatives in this paper.

In a similar way, we can generalize the definitions of constraints, permissions and lions to point-wise operations on streams. For example, for two streams $\alpha = \langle p_0, p_1, p_2, \ldots\rangle$ and $\beta = \langle q_0, q_1, q_2, \ldots\rangle$, we say that α and β are conflicting if there is an index i such that $p_i \wedge q_i$ is inconsistent. Again, the use of streams may lead to new kinds of constraints.

4.1.2 Stronger alternatives for the input/output logic of streams

Alternatively, we can define an input/output logic on streams as if every moment in time, the box is a traditional input/output logic, i.e.,

$$out(G, \langle p_1, p_2, p_3, ...\rangle) = \langle out(G, p_1), out(G, p_2), out(G, p_3), ...\rangle$$

where the first *out* is defined on streams as in Definition 4.1, and the second *out* is a traditional input/output operation defined on propositions.

For a simple example of how the two treatments of streams can give different outputs, consider a component which has $\langle q, q, \overline{\top}\rangle$ amongst the outputs of input $\langle p, p, \overline{\top}\rangle$, and $\langle r, r, \overline{\top}\rangle$ amongst the outputs of input $\langle p, q, \overline{\top}\rangle$, where $\overline{\top}$ is an infinite sequence of tautologies. In the input/output logic of streams of Definition 4.1, we do not necessarily have $\langle q \wedge r, q, \overline{\top}\rangle$ as part of the output of input stream $\langle p, p, \overline{\top}\rangle$. However, this is the case in the stronger alternative defined in the previous paragraph.

The second treatment always gives at least as much output as the first, which follows from a property of the logic of streams as we have defined it. For example, consider the conjunction rule that derives $(a, p \wedge q)$ from (a, p) and (a, q) at every moment in time, then we can also derive the pair of streams $(\langle\overline{a}\rangle, \langle p_0 \wedge q_0, p_1 \wedge q_1, p_2 \wedge q_2, \ldots\rangle)$ from $(\langle\overline{a}\rangle, \langle p_0, p_1, p_2, \ldots\rangle)$ and $(\langle\overline{a}\rangle, \langle q_0, q_1, q_2, \ldots\rangle)$, i.e., we can derive $(\alpha, \beta \wedge \gamma)$ from (α, β) and (α, γ).

As an intermediate solution, we can still define the input/output logic on streams as an input/output operation at every moment in time, but allow for the possibility that the set of norms is updated, i.e.,

$$out(\langle G_1, G_2, G_3, \ldots\rangle, \langle p_1, p_2, p_3, ...\rangle) = \langle out(G_1, p_1), out(G_2, p_2), out(G_3, p_3), ...\rangle$$

This leaves it open how the set of norms can be updated. For example, in the BD architecture in Figure 5 or 6, it is left open how the sets of beliefs and desires are updated after outputs are generated.

4.1.3 Other input/output logics of streams

Thus far, we have assumed integer time, in the sense that the input and output streams may be seen as a function from the natural numbers to propositions. In other words, we have implicitly assumed that there is a clock such that every tick of the clock, a new output is generated from the input. This can be generalized to real time by making time explicit. For example, an abstract behavior type defines an abstract behavior as a relation among a set of timed-data-streams. We do not further consider this extension here.

4.2 Connectors

Thus far we have defined only a single type of channel in a lion connecting the output of a component to the input of another component. However, this is clearly a limitation. Consider, for example, standard Boolean circuits with AND, OR and NOT gates. How does this correspond to lions? For example, in Figure 4 an AND gate only outputs formulas which occur in each case. So if one channel has $Cn(p \wedge q)$ and another channel $Cn(p)$, then the output of the AND gate is $Cn(p)$. Likewise, an OR gate outputs formulae that are in at least one of the input sets, and thus can be used to collect all inputs. Note that under this set-theoretic interpretation, somewhat misleadingly, the logical closure of the output of the OR gate is the consequence set of the conjunction of its inputs. Since the input of a component collects all the outputs of the nodes connected to it, this may implicitly be seen as an OR gate. Moreover, a NOT gate can be defined as a special kind of component. An AND gate seems to extend the expressive power of lions.

Moreover, once we use streams, we can use more complicated channels. A typical example is a register channel, which delays the throughput of data. For example, a register(1) outputs its input with a delay of one clock tick. Such a channel contains a buffer, with an initial value which is outputted as the first element of the output stream.

4.2.1 Register gates

For example, consider the lion with registers in Figure 7. We have a circuit which splits the input, then one direct link and on via register(1). Then it again merges with an OR port. We have the following behavior, assuming that initially there is a tautology in the register:

$$out(\langle p_1, p_2, p_3, \ldots \rangle) = Cn(\langle p_1, p_1 \wedge p_2, p_2 \wedge p_3, \ldots \rangle)$$

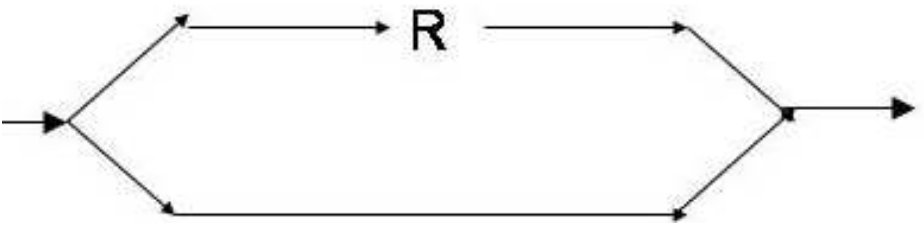

Fig. 7. Connector with register

4.2.2 AND gates

Consider now the extension of lions with AND gates, with the obvious behavior. Moreover, consider the lion in Figure 8, which is the result of replacing the OR gate in Figure 7 by an AND gate. We have the following behavior, if initially there is a contradiction in the buffer:

$$out(\langle p_1, p_2, p_3, \ldots\rangle) = Cn(\langle p_1, p_1 \vee p_2, p_2 \vee p_3, \ldots\rangle)$$

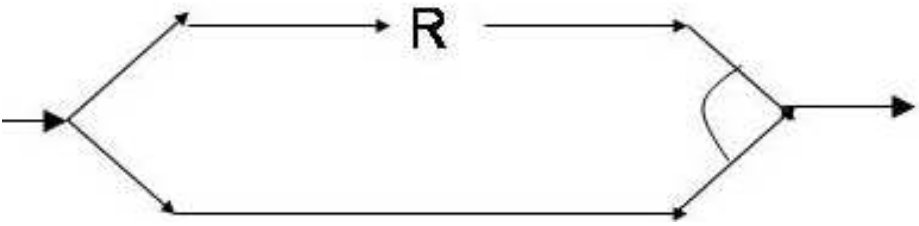

Fig. 8. Connector with register and AND gate

4.2.3 Further extensions

Further inspiration for channels can be found in channel theory [7], based on Dretske's theory of information flow [19], and so-called situation theory, which originated in natural language semantics. Channel theory has studied signalling systems, in which a series of tokens at the physical level, is interpreted as carrying a certain meaning at the semantic level. Dretske defines a communication channel as "that set of existing conditions (on which the signal depends) that either (1) generate no relevant information, or (2) generate redundant information (from the point of view of the receiver) [19, p 115]". In other words: the channel defines a number of background conditions that allow communication, such as for example a shared vocabulary, but it does not add any additional information.

Moreover, several kinds of channels have been studied in the coordination language Reo. Some interesting extensions are the following.

- Channels generally have an input and an output end, but they can also have two input ends, or two output ends, which can be useful to synchronize channels.
- A new kind of gate, called a merger, will accept the input from one of the several channels leading into it, but not all of them at the same time. In the context of lions, such a gate may be called an exclusive choice gate.

- Context dependent channels, for example lossy channels which lose their inputs whenever there is no corresponding take action at the sink side of the channel.

It is an open question which of these channels form a useful extension for lions. As a guidance for further development of lions we propose their use in normative multi-agent systems, which we discuss in the following section.

5 Normative multi-agent systems

In this section we discuss the relevance of interactive computing in general, and lions in particular, for research on agent theory and normative multi-agent systems. A pioneering approach which has inspired us is the work of Jan Treur and colleagues, on the design of interacting reasoning components [33]. Our discussion in this section is based on some of our own research. In the BOID agent architecture, interaction among the various mental attitudes like belief and desire, results in certain types of behavior. In a multi-agent system, a kind of qualitative game theory is used to describe and analyze the interaction among agents. In normative multi-agent systems, norms are used to coordinate the interaction among agents. Finally, in agent communication protocols, interaction is coordinated through social commitments.

5.1 BOID agent architecture

In the BOID agent architecture [13], the behavior of an agent is characterized by the interaction among components that represent the mental attitudes belief, obligation, intention and desire. It extends the BD architecture given in Figure 5 and 6 with components for intentions (I) and obligations (O). The desires, obligations and intentions generate the goals of the agent. Moreover, the BOID architecture extends the BD architecture with the notion of priorities, such that for a social agent, obligations override desires. See [14] for a logical analysis of the interaction among mental attitudes in the BOID architecture.

A challenging issue in the formalization of the BOID architecture is the formalization of planning in this architecture, and in particular the interaction between goal generation and planning [18]. Goal generation is characterized as deductive reasoning and therefore can be formalized as a lion, but planning is characterized as means-end or abductive reasoning, which is less easily formalized as a lion. This issue has been studied using argumentation theory in [20]. Planning has been formalized as a deductive reasoning using so-called practical reasoning rules in the agent programming language 3APL, and an attempt to introduce such practical reasoning rules in the BOID architecture

is given in [17].

A second challenge is the extension of the BOID architecture with streams of inputs and outputs. One central issue is whether mental attitudes persist over time, an issue discussed for example by the commitment strategies in the BDI literature.

5.2 Multi-agent systems

Interaction plays a central role in the design and analysis of multi-agent systems. In particular, the interaction is often based on game-theoretic approaches borrowed from economics and bounded reasoning as studied in artificial intelligence. In these approaches, an agent itself can be seen as a black box, with the agent's observations as input, and the actions that constitute its behavior as output. When an abstract behavior type characterizes a certain type of agent behavior, it is referred to as an agent type [13].

Whereas classical game theory is based on an equilibrium analysis, theories in artificial intelligence and agent theories are often based on the notion of recursive modeling. Agents model other agents when making decisions, but the number of levels in such recursive models is finite. Equilibrium analysis can be seen as reasoning about an infinite number of levels, which conflicts with the idea of bounded rationality.

5.3 Normative multi-agent systems

In normative multi-agent systems, norms are used to coordinate the interaction among intelligent agents. Various kinds of norms have been studied, such as regulative norms, permissive norms, constitutive norms, social laws, etc. Norms are modeled in terms of interaction, but the set of norms or normative system is often modeled as a component, interacting with the agents. The Boella-van der Torre model of normative multi-agent systems [9] is based on a more general notion of agent than is usually understood. This model makes it possible to attribute mental attitudes to normative systems, which means that the normative system is itself modeled as an agent – it is called a socially constructed agent. This has two important advantages:

(i) Obligations can be defined in the BDI framework. The desires or goals of the normative system are the obligations of the agent. This contributes to the open problem whether norms and obligations should be represented explicitly, for example in a deontic logic, or they can also be represented implicitly.

(ii) The interaction between an agent and the normative system can be modeled as a game between two agents. Consequently, methods and tools

used in game theory such as equilibrium analysis can be applied to normative reasoning.

For example, a qualitative game theory can be developed as follows. The bearer of the obligation models the decision making of the normative system, which may itself be based on a model of the bearer of the norm. Hence the term recursive modeling. In so called violation games, the agent bases its decision on the consequences of the normative system's anticipated reaction, using a profile of the system's beliefs, desires and goals. In particular, the agent is interested in the question whether the normative system will consider its decision as a violation of the norm and thus sanction the agent. So called recursive games add one level of recursion to the game. They can be used by a legislator, to decide which norms should be created, or what the level of sanction should be. For example, in contract negotiation games, agents recursively model other agents to find out whether they would obey the new conditions and norms of the contract or not.

5.4 Norms

Regulative norms like obligations can be defined as follows. If agent A is obliged to make sure that a, then agent N may decide that the absence of a counts as a violation of some norm n and that agent A must be sanctioned. In particular, this means that

(i) Agent A believes that agent N desires that A does a (objective)

(ii) Agent A believes that agent N desires $\neg V(n)$, that there is no violation of norm n, but if agent N believes $\neg a$ then N has the goal $V(n)$, $\neg a$ counts as a violation (detection).

(iii) Agent A believes that agent N desires $\neg s$, not to sanction, but if agent N decides $V(n)$ then it has as a goal that it sanctions agent A by doing s. Agent N only sanctions in case of violation. Moreover, agent A believes that agent N has a way to apply the sanction (sanctioning)

(iv) Agent A desires $\neg s$: it does not like the sanction (deterrence)

The clauses deal with various circumstances. For example, the first clause expresses the objective of the norm. The second clause ensures that normative systems will only detect actual violations. Similarly, the third clause prevents arbitrary sanctions. The last clause shows that the sanction is undesirable, and hence will deter agents from violating the norm.

We can define different agent types. For example, respectful agents simply desire to fulfill the norm as such, whereas for selfish agents the only motivation to comply with obligations is the fear for sanction or the desire for reward.

Intermediate agent types are possible too. A selfish agent exploits the beliefs and goals of the normative system to try and violate an obligation without being detected or sanctioned.

In addition to regulative norms, we stress the importance of constitutive norms [31]. Just as the rules of chess constitute the game by defining legal moves, constitutive norms generate institutional facts, that allow individuals to function in society. Constitutive norms take the form of 'counts as' rules. For example, a theater ticket counts as evidence of the right to occupy a seat. Constitutive norms apply only under certain circumstances and are intimately linked to an authority or institution. Thus constitutive rules are of the form "event or fact x counts as event or fact y under circumstances c in institution i".

Coordination in normative multi-agent systems [9] uses besides norms also more complex concepts like contracts [11] and roles [10].

5.4.1 Contracts

Coordination can be achieved without requiring a fixed agent architecture, but by means of roles and norms as incentives for cooperation. For this purpose, some approaches allow agents to stipulate contracts. A contract can be defined as a statement of intent that regulates behavior among organizations and individuals.

Contracts have been proposed to make explicit the way agents can change the interaction with and within the society: they create new obligations or permissions and new possibilities of interaction. From a contractual perspective, an organization can be seen as the possible set of agreements for satisfying the diverse interests of self interested individuals. In [11] we define a contract as a triple:

(i) An institutional fact $c \in I$ representing that the contract has been created.

(ii) A constitutive rule which makes the institutional fact c true.

(iii) A set of constitutive rules having as antecedent the creation c of the contract and as consequent creation actions modifying the mental attitudes of the organization.

5.4.2 Roles

Multi-agent systems are often proposed as a solution for the organizational design of open systems. A key notion in the structure of an organization is that of role. Roles specify the activities of an organization to achieve its overall

goal, while abstracting from the individuals that will eventually execute them. A role is usually described in terms of normative descriptions, expectations, standardized patterns of behavior, social commitments, goals and planning rules. The normative description specifies the obligations that any agent who plays the role (called the actor) should obey. Goals are his intrinsic motivations. Roles, thus, seem to be strictly related to the notion of agent: they are described using notions like actions, goals and obligations.

In the Boella-van der Torre model, we attribute mental attitudes to roles and use this model for several problems. First, if we design an organization in terms of roles by attributing mental attitudes to them, we can design it just as we would design an organization in terms of agents. However, the organization does not depend on the agents operating in it: agents are replacable. Tasks can be delegated to roles, with a greater flexibility: it would be not sufficient to describe roles as machines. Second, if we consider the problem of assigning roles to actors and we attribute mental attitudes to roles, then the problem becomes finding a match between the beliefs representing expertise of the role and the beliefs of the actor. Third, if we consider the governance of organizations, then the attribution of mental attitudes to roles makes it easier to verify whether an agent is acting according to the overall plan of the organization: if he is acting according to the role's beliefs and achieving the role's goals, then he is fulfilling his responsibilities.

The issue discussed in the formalization is which beliefs and goals are attributed to a role. Not all the beliefs of the organization are beliefs of a role due to the role assignment problem: the less beliefs are ascribed to a role, the easier it becomes to assign agents to this role. Similarly for goals – not all the goals of an organization are the goals of a role due to the governance problem: to distribute responsibilities over the agents, and to control the running system. Consequently, the organizational design aims at defining roles with minimal sets of beliefs and goals. Note that not only the beliefs and goals of an organization may not correspond to goals of the role, but also vice versa: a goal of a role may not be a goal of the organization.

5.5 Communication

In the communication of computer systems, interaction is regulated by a protocol: a set of rules that define which messages are allowed by which participants at which stage of the interaction. Well known examples are the Contract Net Protocol, that regulates a distributed procedure for allocating tasks, or protocols for negotiation. Protocols have a normative aspect: they consist of rules specifying what an agent should and shouldn't do (prescription); based on those rules other agents can form expectations about the behavior of other

participants (prediction).

In agent theory there are at least two kinds of semantics. One is based on the paradigm of mental attitudes. it makes specific assumptions about the internal architecture of the agents. The other is based on social commitments, and makes no assumptions about the agents. This approach is more in line with the paradigm of interactive computing. In this section we discuss the two approaches, and also propose a synthesis [8].

5.5.1 Speech act theory

Within the autonomous agents and multi-agent systems community, agent communication standards have been developed for structuring messages and for simple interaction protocols. The backbone of these standards is formed by speech act theory [6,30]. The semantics of a speech act is commonly given by the preconditions and intended effect on the mental state of an agent, expressed using modal operators for belief and intention. Early research in AI [16] showed that the systematic analysis of speech acts in terms of preconditions and postconditions could be modeled straightforwardly in terms of planning operators. That means that a sequence of messages makes sense, in case it can be interpreted as a plan to achieve some goal.

Various aspects of speech act theory for agents have been standardized by an organization called FIPA, and this standardization effort has been a relative success, although it has been criticized heavily. A point of criticism is that it is impossible to verify the correct usage of a speech act, since for most realistic multi-agent settings the mental state of an agent is inaccessible. Agents may well be lying. So unless a sincerity principle is postulated – which makes no sense under common assumptions regarding multi-agent systems – it is impossible to verify protocols [36]. What is needed instead is a semantics that is based on *public information* about what agents are committed to, on the basis of what they have said.

5.5.2 Social semantics

For this reason, Singh [32] proposes a social semantics. It is based on the notion of *commitment*. Commitment binds a speaker to the community and, unlike mental attitudes, commitments have a public character. In some cases commitments generate a kind of obligation. For example, when a speaker asserts a proposition, the speaker generally becomes committed to subsequently defending the proposition, when challenged by another dialogue participant.

These two approaches – mentalistic and social – are portrayed as competing alternatives, but it is quite possible to combine them. The combination rests

on the following key idea: if we attribute mental attitudes to the roles of the participants, rather than to the individual agents that enact the roles, the mental attitudes do get a public character, just like commitments.

Such a combination has the following advantages. On the one hand applications which take place in a restricted known cooperative environment can re-use much of the work done on FIPA compliant agent communication. On the other hand, stronger obligations can be added when required. In particular, in competitive or other non-cooperative circumstances, no sincerity principle needs to be assumed. Instead an obligation to defend a publicly attributed belief, as suggested above, must be added. However, the basic bookkeeping and the semantics of the speech acts themselves, can remain the same.

5.5.3 Synthesis

The method we adopt is to model dialogue as a game in which agents play roles. Speech acts are moves in the game and their preconditions and effects refer to the mental states attributed to the roles, not to the mental states of the agents themselves. This approach presupposes that mental attitudes can be attributed to roles as well as to agents. Following [10,12], we describe roles as agents with mental attitudes, albeit of a different kind, since they are not autonomous.

Agents playing roles in the game can affect the state of the game, the state of the roles they play and the state of the other roles. This is possible thanks to constitutive rules. Constitutive rules explain how an utterance by a player counts as a move in the game and how the move affects the beliefs and goals of the roles. In contrast with agents' beliefs and goals, beliefs and goals of roles have a public character in that the game defines them, and it also defines how they evolve during a dialogue according to the moves played by the agents. Moreover, since roles provide a description of expected behavior they have a prescriptive character: the player of a role becomes committed to the beliefs and goals of the role created during the dialogue. Finally, the constitutive rules can describe, if necessary, how obligations are created during the dialogue.

Since such dialogue games consist of both constitutive and regulative rules, they can be considered as a normative multi-agent system. Just as for roles, our view of normative systems is based on the agent metaphor: a normative system can technically be considered as an agent to which mental attitudes are attributed. Moreover, normative systems, like organizations, can be articulated in roles.

5.5.4 Example

1. s:	The president will win the election.	$\mathit{inform}(s, r, p)$
2. r:	But there is fraud,	$\mathrm{inform}(r, s, q)$
	so the president will not win.	$\mathrm{inform}(r, s, (q \supset \neg p))$
3. s:	Fraud? But why!	$\mathrm{challenge}(s, r, q)$
4. r:	Fair enough, no fraud.	$\mathrm{retract}(r, s, \mathrm{inform}(r, s, q))$
	So you're right.	$\mathrm{accept}(r, s, p)$

The example is explained as follows.

Turn 1: role s informs r that p. The rules of role s now state that p is a consequence of s's beliefs. The rules of role r state that the belief p can be attributed to r unless r challenges s's inform.

Turn 2: the agent playing role r challenges s's information by means of an argument of the form $q, (q \supset \neg p)$.

Turn 3: s is risking losing the game. Unless it does something, it will get into a contradiction, since r's argument supports $\neg p$, which is in contrast with its current beliefs (p). It has some alternatives: retracting its first *inform* or challenging r's argument. So s decides to challenge r's challenge by asking for justification (*why*).

Turn 4: r retracts the *inform* about q, thus giving up its challenge to p, and subsequently accepts p.

This example shows that indeed utterances can be modeled as constitutive rules, with the preconditions and postconditions of the mentalistic approach, reinterpreted as the mental attitudes of the roles.

6 Summary

In this paper we consider logical input/output nets, or lions for short, as models of interactive computing. Lions are based on input/output logic, a deontic logic not used as a non-classical inference engine deriving output from input, but as a secretarial assistant for transformations from input to output. We present a few definitions, but no formal results.

The central research question of this paper is how notions developed in interactive computing such as abstract behavior types, the coordination language Reo, and Boolean circuits with registers, can be used to extend input/output logics and lions. In particular, we consider two extensions of input/output logics and lions.

First, we consider input/output logics defined on infinite sequences (or streams) of inputs and outputs. The first extension replaces the propositional logic underlying input/output logic by a logic of streams, and the second extension strengthens this logic by applying the input/output logic on every

input. Yet another alternative applies an input/output operation at every propositional input, but assumes that the set of norms can change over time.

Secondly, we consider lions with AND and register gates, formalizing the behavior of channels and connectors. The extension illustrates how gates from Boolean circuits can be used to extend lions, and how operators for streams can be introduced to model behavior over time. We also observe that more complicated channels can be introduced, such as lossy channels.

Our discussion has highlighted a number of future research directions for lions. A proof theory for lions can be developed, constraints can be added to lions, and permissions can be introduced to distinguish two kinds of outputs. The introduction of time in streams introduces new possibility of studying new proof rules of input/output logics, and the introduction of new kinds of channels and gates introduces a variety of new conceptual and formal issues.

Given the huge number of possible extensions, we feel that we need guidance to develops lions, just like the development of input/output logic has been guided by issues in deontic logic. We are motivated to build normative multi-agent systems on top of lions, based on the idea that interaction plays a crucial role in normative multi-agent systems. In the agent architecture the behavior of an agent is determined by the interaction among the mental attitudes, in a multi-agent system game theory is used to describe and analyze the interaction among agents, in normative multi-agent systems norms are used to coordinate interactions among agents, and in communication interaction is central in models based on social commitments.

One particular extension we believe can be guiding the further development of lions for normative multi-agent systems is the development of value webs, which are networks that describe the exchange of value. For example, a value web may model the exchange of goods for money, possibly including third parties such as shippers and banks. We suspect that when not only information but also values are transported over channels, new issues arise.

We also believe that the research on input/output logics, lions and normative multi-agent systems can be an inspiration for the wider study of interactive computing. In particular, we like to emphasize three issues:

Logic The use of logic as a secretarial assistant, which facilitates exogenous coordination.

Concepts Concepts of normative multi-agent systems to study interaction like contracts and roles.

Communication Communication based on social commitments, and its relation to the mentalistic FIPA approach.

Acknowledgements

Thanks to David Makinson for the introduction of logical input/output nets (personal communications) and for commenting on an earlier version of this paper, and to Farhad Arbab and his colleagues at CWI for many inspiring discussions on the issues raised in this paper.

References

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

[2] Arbab, F., *What do you mean, coordination?*, Bulletin of the Dutch Association for Theoretical Computer Science, NVTI (1998), pp. 11–22, available on-line http://www.cwi.nl/NVTI/Nieuwsbrief/nieuwsbrief.html.

[3] Arbab, F., *Reo: A channel-based coordination model for component composition*, Mathematical Structures in Computer Science **14** (2004), pp. 329–366.

[4] Arbab, F., *Abstract behavior types: A foundation model for components and their composition*, Science of Computer Programming **55** (2005), pp. 3–52.

[5] Arbab, F., *Coordination of interacting concurrent computations*, in: *Interactive Computation: The New Paradigm*, Springer, 2005 .

[6] Austin, J. L., "How to Do Things with Words," Harvard University Press, Cambridge, Mass, 1962.

[7] Barwise, J. and J. Seligman, "Information Flow," Cambridge University Press, 1997.

[8] Boella, G., J. Hulstijn and L. van der Torre, *A synthesis between mental attitudes and social commitments in agent communication languages*, in: *Proceedings of IAT 2005* (2005).

[9] Boella, G. and L. van der Torre, *Attributing mental attitudes to normative systems*, in: *Proceedings of the second international joint conference on autonomous agents and multi agent systems (AAMAS'03)*, 2003, pp. 942–943.

[10] Boella, G. and L. van der Torre, *Attributing mental attitudes to roles: The agent metaphor applied to organizational design*, in: *Procs. of ICEC'04* (2004), pp. 130–137.

[11] Boella, G. and L. van der Torre, *Contracts as legal institutions in organizations of autonomous agents*, in: *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS'04)*, 2004, pp. 948–955.

[12] Boella, G. and L. van der Torre, *Regulative and constitutive norms in normative multiagent systems*, in: *Procs. of 9th International Conference on the Principles of Knowledge Representation and Reasoning*, 2004, pp. 255–265.

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

[14] Broersen, J., M. Dastani and L. van der Torre, *Realistic desires*, Journal of Applied Non-Classical Logics **12(2)** (2002), pp. 287–308.

[15] Broersen, J., M. Dastani and L. van der Torre, *Beliefs, obligations, intension and desires as components in agent architectures*, International Journal of Intelligent Systems (2005).

[16] Cohen, P. and H. Levesque, *Intention is choice with commitment*, Artificial Intelligence **42 (2-3)** (1990), pp. 213–261.

[17] Dastani, M. and L. van der Torre, *Programming BOID agents: a deliberation language for conflicts between mental attitudes and plans*, in: *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS'04)*, 2004, pp. 706–713.

[18] Dastani, M. and L. van der Torre, *What is a normative goal? towards goal-based normative agent architectures*, in: *Postproceedings RASTA02*, LNAI **2934**, Springer, 2004 pp. 210–227.

[19] Dretske, F., "Knowledge and the Flow of Information," MIT Press, 1981.

[20] Hulstijn, J. and L. van der Torre, *Combining goal generation and planning in an argumentation framework*, in: *Proceedings NMR'04*, 2004, pp. 212–218.

[21] Jones, A. and J. Carmo, *Deontic logic and contrary-to-duties*, in: D. Gabbay, editor, *Handbook of Philosophical Logic*, Kluwer, 2002 pp. 203–279.

[22] Makinson, D. and L. van der Torre, *Input-output logics*, Journal of Philosophical Logic **29** (2000), pp. 383–408.

[23] Makinson, D. and L. van der Torre, *Constraints for input-output logics*, Journal of Philosophical Logic **30** (2001), pp. 155–185.

[24] Makinson, D. and L. van der Torre, *Permissions from an input/output perspective*, Journal of Philosophical Logic **32 (4)** (2003), pp. 391–416.

[25] Makinson, D. and L. van der Torre, *What is input/output logic?*, in: *Foundations of the Formal Sciences II: Applications of Mathematical Logic in Philosophy and Linguistics*, Trends in Logic **17**, Kluwer, 2003 .

[26] Poole, D., *A logical framework for default reasoning*, Artificial Intelligence **36** (1988), pp. 27–47.

[27] Reiter, R., *A logic for default reasoning*, Artificial Intelligence **13** (1980), pp. 81–132.

[28] Rutten, J., *Algebra, bitstreams and circuits*, Technical Report SEN-R0502, CWI Amsterdam (2005).

[29] Savage, L., "The foundations of statistics," John Wiley & Sons, New York, 1954.

[30] Searle, J., "Speech Acts: an Essay in the Philosophy of Language," Cambridge University Press, Cambridge (UK), 1969.

[31] Searle, J., "The Construction of Social Reality," The Free Press, New York, 1995.

[32] Singh, M. P., *A social semantics for agent communication languages*, in: F. Dignum and M. Greaves, editors, *Issues in Agent Communication: Proceedings of the IJCAI Workshop on Agent Communication Languages*, LNCS 1916, Springer, 2000 pp. 31 – 45.

[33] Treur, J., *Semantic formalisation of interactive reasoning functionality*, International Journal of Intelligent Systems **17** (2002), pp. 645–686.

[34] Wegner, P., *Why interaction is more powerful than algorithms*, Communications of the ACM **40** (1997), pp. 80 – 91.

[35] Wieringa, R. and J.-J. Meyer, *Applications of deontic logic in computer science: A concise overview*, in: J.-J. Meyer and R. Wieringa, editors, *Deontic Logic in Computer Science: Normative System Specification*, John Wiley & Sons, Chichester, England, 1993 pp. 17–40.

[36] Wooldridge, M. J., *Semantic issues in the verification of agent communication languages*, Journal of Autonomous Agents and Multi-Agent Systems **3** (2000), pp. 9–31.