ESSLLI08: Deontic Logic in Computer Science Part 2a/5: The Input/Output Logic Framework

Jörg Hansen and Leendert van der Torre

Makinson and van der Torre see a normative code as a set G of conditional norms, which is a set of ordered pairs (a, x) [5]. 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. Moreover, given any universe L such that $G \subseteq L^2$ and an input $A \subseteq L$, they suggest that the output of A under G may be understood simply as

$$G(A) = \{x \mid (a, x) \in G \text{ for some } a \in A\}$$

Input/output logic investigates what happens to this basic picture when we pass to the logical level, i.e. when L is a propositional language, closed under at least the usual truth-functional connectives, and G a set of ordered pairs (a, x) of formulae in L. Since this investigation is relevant not only for deontic logic, Makinson and van der Torre refer to G not as a normative code, but as a generating set. To avoid all confusion, the generators G are not treated as formulae, but simply as ordered pairs (a, x) of purely boolean (or eventually first-order) formulae. They read a pair (a, x) forwards, i.e. with a as body and x as head; and they call the corresponding truth-functional formula $a \to x$ its materialization, echoing the old name material implication for the connective involved.

Suppose that we are also given a set A of formulae. The problem studied in input/output logic is: how may we reasonably define the set of propositions x making up the output of A under G, or one might also say, of G given A, which we write *out*(G, A)? Makinson and van der Torre emphasize that 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'.

1 Obligations

The basic intuition is that input and output are both under the sway of the operation Cn of classical consequence. Makinson and van der Torre's simplest response to their problem is to put

$$out(G, A) = Cn(G(Cn(A)))$$

where the function $G(.)$ is defined as on the pre-logical level above, and Cn alias $\vdash$ is classical consequence. In other words, given a set A of formulae as input, they first collect all of its consequences, then apply G to them, and finally consider all of the consequences of what is thus obtained. They also define various variants to deal with disjunctive inputs intelligently, and making outputs available for recycling as inputs.

Definition 1 (Obligations) *[2] Let L be a propositional logic with $\top$ a tautology, and let G be a set of ordered pairs of L (called the generators). A generator (a, x) is read as 'if input a then output x'. The following logical systems have been defined, where v ranges over boolean valuations or the function that puts $v(b) = 1$ for all formulae b, and $V = \{b \mid v(b) = 1\}$.*

$$\begin{aligned}
out_1(G, A) &= Cn(G(Cn(A))),\\
out_2(G, A) &= \cap\{Cn(G(V)) \mid v(A) = 1\},\\
out_3(G, A) &= \cap\{Cn(G(B)) \mid A \subseteq B = Cn(B) \supseteq G(B)\},\\
out_4(G, A) &= \cap\{Cn(G(V)) : v(A) = 1 \text{ and } G(V) \subseteq V\}.
\end{aligned}$$

The following example illustrates and compares the four input/output logics. The most characteristic property is that inputs are not in general outputs; that is, we do not have $A \subseteq out_1(G, A)$.

Example 1 *[2] Put generators $G = \{(a, x), (b, x), (x, y)\}$, where a, b, x and y are distinct elementary letters, and put $A = \{a\}$. Inputs are not in general outputs, since $G(Cn(a)) = \{x\}$ so $a \notin out_1(G, a) = Cn(G(Cn(a))) = Cn(x)$. Contraposition also fails, for although $x \in out_1(G, a)$ we have $\neg a \notin out_1(G, \neg x)$: since $a \in Cn(\neg x)$ we have $G(Cn(\neg x)) = \emptyset$ so that $\neg a \notin out_1(G, \neg x) = Cn(G(Cn(\neg x))) = Cn(\emptyset)$.*

We do not have $y \in out_1(G, A)$. However, in certain situations, it may be appropriate for outputs to be available for recycling as inputs. For example, the elements (a, x) of G may be conditional norms of a kind that say that any configuration in which a is true is one in which x is desirable. In some contexts, we may wish to entertain hypothetically the items already seen as desirable, in order to determine what is in turn so. We do have $y \in out_3(G, A)$ and $y \in out_4(G, A)$.

Finally, put $A = \{a \vee b\}$. Then $Cn(A) \cap b(G) = \emptyset$ where we write $b(G)$ for the set of all bodies of elements of G, i.e. in this example the set $\{a, b\}$. Hence also $G(Cn(A)) = \emptyset$ so that $out_1(G, A) = Cn(G(Cn(A))) = Cn(\emptyset)$. However, in many contexts we would want to put x in the output, as it can be obtained from each of the two disjuncts of the input. We do have $x \in out_2(G, A)$ and $x \in out_4(G, A)$.

Input/output logic is axiomatized as a kind of conditional logic, where one is used to ask the following question. Suppose we are given only the generating set G: how may we define the set of input/output pairs (A, x) arising from G, written as *out*(G)? Makinson and van der Torre suggest that this is the same question as asking what is *out*(G, A), because they define $(A, x) \in out(G)$ iff

$x \in out(G, A)$ or conversely. They also suggest that the two formulations give a rather different gestalt, and one is sometimes more convenient rather than the other. Whereas the latter tends to be clearer in semantic contexts, the former is easier to work with when considering derivations in a syntactic context. They move freely from one to the other, just as one moves between Cn and $\vdash$ for classical consequence.

Theorem 1 *[2] Let L be a base logic with $\top$ a tautology, and let G be a set of ordered pairs of L (called the generators). Input/output logic out_1 (out_2 / out_3 / out_4) is a closure operation on $G \cup \{(\top, \top)\}$ under replacement of logical equivalents and the rules SI, WO and AND (and OR / CT / OR and CT).*

SI	$\frac{(a,x)}{(a\wedge b,x)}$	*WO*	$\frac{(a,x)}{(a,x\vee y)}$	*AND*	$\frac{(a,x),(a,y)}{(a,x\wedge y)}$
OR	$\frac{(a,x),(b,x)}{(a\vee b,x)}$	*CT*	$\frac{(a,x),(a\wedge x,y)}{(a,y)}$	*ID*	$\frac{}{(a,a)}$

Example 2 *Given $G = \{(a, x), (a, y), (x, z)\}$ the output of G contains $(a \wedge b, x)$, $(a \wedge x, z)$, $(a, x \vee y)$, $(a, a \vee x)$, and $(a, x \wedge y)$ using rules SI, WO and AND. Using also the CT rule, the output contains (a, z).*

2 Permissions

Permissions are more ambiguous than obligations, and various notions have been defined. Makinson and van der Torre [4] distinguish three notions of permission. First, *negperm* is the negation of an prohibition, it corresponds to what is called weak permission. Second, *statperm* guides the citizen in the deontic assessment of specific actions, and behaves like a weakened obligation: given what is obligatory and what is strongly permitted the actual permissions of an agent are computed. If P is the set of permissive norms, then we have $statperm(P, G) \subseteq out(P \cup G)$, see [4] for details. Third, *dynperm* guides the legislator by describing the limits on what may be prohibited without violating static permissions, which is called prohibition immunity: "on the other hand, dynamic permission corresponds to 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 has been positively permitted in a certain realizable situation, then adding the prohibition is inadmissible under pain of a certain kind of incoherence, and the pair (a, x) is to that extent immune from prohibition. For this reason, dynamic permission could also be called prohibition immunity" [4].

Definition 2 (Permissions) *Let G and P be two sets of generators, where P stands for permissive norms, and let out be an input/output logic.*

- *$(a, x) \in negperm(G)$ iff $(a, \neg x) \notin out(G)$;*
- *$(a, x) \in statperm(P, G)$ iff $(a, x) \in out(G \cup Q)$ for some singleton or empty $Q \subseteq P$;*
- *$(a, x) \in dynperm(P, G)$ iff $(c, \neg z) \in out(G \cup \{(a, \neg x)\})$ for some pair $(c, z) \in statperm(P, G)$ with c consistent.*

Example 3 *It is obligatory to make homework, but if one does homework he is permitted to watch the television, $G = \{(\top, h)\}$ and $P = \{(h, w)\}$. Then $(\top, h) \in negperm(G)$, since what is obligatory is permitted and $(a, b) \in negperm(G)$ since given a there is no restriction about b. Moreover, $(h, w) \in statperm(P, G)$ since this is explicitly permitted and $(a, w) \in dynperm(P, G)$: $(a \wedge h, \neg w) \in out(G \cup \{(a, \neg w)\})$ for some pair $(a \wedge h, w) \in statperm(P, G)$.*

3 Constraints

The main problem of reasoning with obligations and permissions is the question how to deal with violations and obligations resulting from violations, known as contrary-to-duty reasoning. It has been discussed in the context of the notorious contrary-to-duty paradoxes such as Chisholm's and Forrester's paradox. It has led to the use of constraints in input/output logics [3].

The 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, input/output logic carries out a contraction on the set G of generators. In case of contrary-to-duty obligations, the input represents something which is inalterably true, and an agent has to ask himself which obligations (output) this input gives rise to: even if the input should have not come true, an agent has to "make the best out of the sad circumstances" [1].

In input/output logics under constraints, a set of generators and an input does not have a set of propositions as output, but a set of sets of propositions. We can infer a set of propositions by for example taking the join (credulous) or meet (sceptical), or something more complicated. Besides, we can adopt an output constraint (the output has to be consistent) or an input/output constraint (the output has to be consistent with the input). In this handout we consider only the input/output constraints.

Definition 3 (Constraints) *Let G be a set of generators and out be an input/output logic. Moreover, we write $x \in out(G, a)$ iff $(a, x) \in out(G)$. We define:*

- *$maxfamily(G, a)$ is the set of $\subseteq$-maximal subsets G' of G such that $out(G', a) \cup \{a\}$ is consistent.*
- *$outfamily(G, a)$ is the output under the elements of maxfamily, i.e., $\{out(G', a) \mid G' \in maxfamily(G, a)\}$.*
- *$(a, x) \in out_\cup(G)$ iff $x \in \cup outfamily(G, a)$*
  $(a, x) \in out_\cap(G)$ iff $x \in \cap outfamily(G, a)$

Makinson and van der Torre [3] consider the following example.

Example 4 *Multiple level of violation may be analyzed. For example, put $G = \{(\top, \neg a), (a, x), (a \wedge \neg x, y)\}$ where a is read as 'you break your promise', x as 'you apologize' and y as 'you are ashamed'. Consider the input $a \wedge \neg x$. On the one hand, $out(G, a \wedge \neg x) = Cn(\neg a, x, y)$, which is consistent. On the other hand, $out(G, a \wedge \neg x)$ is inconsistent with input $a \wedge \neg x$, so that $maxfamily(G, a \wedge \neg x) = \{(a \wedge \neg x, y)\}$ and $outfamily(G, a \wedge \neg x) = \{Cn(y)\}$.*

Permissions under constraints can be formalized by replacing in Definition 2 each occurrence of *out* by $out_\cup$ or $out_\cap$.

REFERENCES

[1] B. Hansson, 'An analysis of some deontic logics', *Noûs*, **3**, 373–398, (1969).

[2] D. Makinson and L. van der Torre, 'Input-output logics', *Journal of Philosophical Logic*, **29**(4), 383–408, (2000).

[3] D. Makinson and L. van der Torre, 'Constraints for input-output logics', *Journal of Philosophical Logic*, **30(2)**, 155–185, (2001).

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

[5] D. Makinson 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*, volume 17 of *Trends in Logic*, Kluwer, (2003).