What is Input/Output Logic? Input/Output Logic, Constraints, Permissions*

David Makinson[1] and Leendert van der Torre[2]

[1] david.makinson@googlemail.com
[2] University of Luxembourg, Computer Science and Communications (CSC)
1359, Luxembourg, 6 rue Richard Coudenhove Kalergi, Luxembourg
leendert@vandertorre.com

Abstract. We explain the *raison d'être* and basic ideas of input/output logic, sketching the central elements with pointers to other publications for detailed developments. The motivation comes from the logic of norms. Unconstrained input/output operations are straightforward to define, with relatively simple behaviour, but ignore the subtleties of contrary-to-duty norms. To deal with these more sensitively, we constrain input/output operations by means of consistency conditions, expressed via the concept of an outfamily. They also provide a convenient platform for distinguishing and analysing several different kinds of permission.

Keywords. Deontic logic, input/output logic, constraints, permissions

1 Motivation

Input/output logic takes its origin in the study of conditional norms. These may express desired features of a situation, obligations under some legal, moral or practical code, goals, contingency plans, advice, etc. Typically they may be expressed in terms like: *In such-and-such a situation, so-and-so should be the case*, or *. . . should be brought about*, or *. . . should be worked towards*, or *. . . should be followed* – these locutions corresponding roughly to the kinds of norm mentioned.

To be more accurate, input/output logic has its source in a tension between the philosophy of norms and formal work of deontic logicians.

Philosophically, it is widely accepted that a distinction may be drawn between norms on the one hand, and declarative statements on the other. Declarative statements may bear truth-values, in other words are capable of being true or false; but norms are items of another kind. They may be respected (or not), and may also be assessed from the standpoint of other norms, for example when a legal norm is judged from a moral point of view (or vice versa). But it makes no sense to describe norms as true or as false.

However the formal work of deontic logicians often goes on as if such a distinction had never been heard of. The usual presentations of deontic logic, whether

* This paper extends [11] with Section 6 on permissions.

axiomatic or semantic, treat norms as if they could bear truth-values. In particular, the truth-functional connectives *and*, *or* and most spectacularly *not* are routinely applied to norms, forming compound norms out of elementary ones. Semantic constructions using possible worlds go further by offering rules to determine, in a model, the truth-value of a norm.

This anomaly was noticed more than half a century ago, by Dubislav [4] and Jørgensen [5], but little was done about it. Indeed, from the 1960s onwards, the semantic approach in terms of possible worlds deepened the gap. The first serious attempt by a logician to face the problem appears to be due to Stenius [15], followed by Alchourrón and Bulygin [2] for unconditional norms, then Alchourrón [1] and Makinson [7] for conditional ones. Input/output logic may be seen as an attempt to extract the essential mathematical structure behind these reconstructions of deontic logic.

Like every other approach to deontic logic, input/output logic must face the problem of accounting adequately for the behaviour of what are called 'contrary-to-duty' norms. The problem may be stated thus: given a set of norms to be applied, how should we determine which obligations are operative in a situation that already violates some among them? It appears that input/output logic provides a convenient platform for dealing with this problem by imposing consistency constraints on the generation of output.

We begin by outlining the central ideas and constructions of unconstrained input/output logic. These are quite straightforward, and provide the basic framework of the theory. We then sketch a strategy for constraining those operations so as to deal more sensitively with contrary-to-duty situations. Finally, we explain how the same operations may be deployed in the analysis of permission.

For further details, the reader is invited to refer to Makinson and van der Torre [8,9].

2 Unconstrained Input/Output Operations

We avoid assuming that conditional norms bear truth-values. They are not embedded in compound formulae using truth-functional connectives. To avoid all confusion, they are not even treated as formulae, but simply as ordered pairs (a, x) of purely boolean (or eventually first-order) formulae.

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'.

The simplest kind of unconstrained input/output operation is depicted in Figure 1. 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

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

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.*

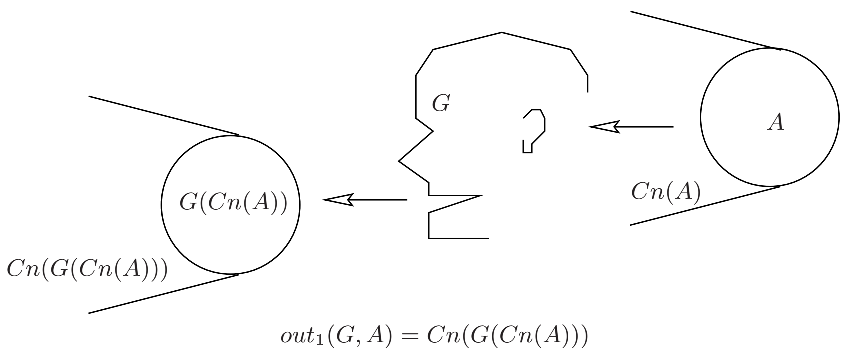


Fig. 1. Simple-minded Output

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 $out_1(G, a)$. It 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:

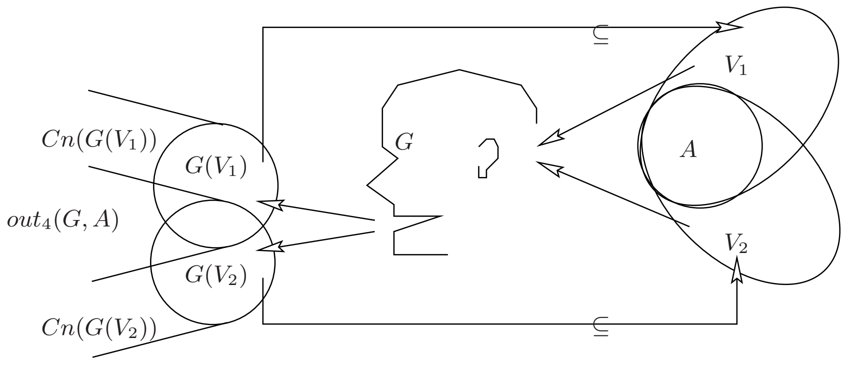
Strengthening Input (SI):	From (a, x) to (b, x) whenever $a \in Cn(b)$
Conjoining Output (AND):	From (a, x), (a, y) to $(a, x \wedge y)$
Weakening Output (WO):	From (a, x) to (a, y) whenever $y \in Cn(x)$

But simple-minded output lacks certain 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. By classical logic, if $x \in Cn(a)$ and $x \in Cn(b)$ then $x \in Cn(a \vee b)$. But there is nothing to tell us that if $x \in out_1(G, a) = Cn(G(Cn(a)))$ and $x \in out_1(G, b) = Cn(G(Cn(b)))$ then $x \in out_1(G, a \vee b) = Cn(G(Cn(a \vee b)))$.

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, pictured by diagrams in the same spirit as that for simple-minded output, and 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).

For example, reusable basic output may be given a diagram and definition as in Figure 2. In the definition, a complete set is one that is either maximally consistent or equal to the set of all formulae.



$$out_4(G, A) = \cap\{Cn(G(V)) \mid A \subseteq V \supseteq G(V), V \text{ complete}\}$$

Fig. 2. Basic Reusable Output

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). Diagrammatically it is to add arrows from G's ear to mouth. 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 may be shown directly in terms of the definitions, or using the following simple derivation of CT from the other rules.

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

The application of WO here is justified by the fact that we have $y \in Cn(x \wedge (a \wedge \neg x))$ since the right hand formula is a contradiction. Note that all rules available in basic throughput (including, in particular, identity) are needed in the derivation, reflecting the fact that CT is not derivable in the weaker systems.

This strong system indeed collapses into classical consequence, in the sense that $out_4^+(G, A) = Cn(m(G) \cup A)$ where $m(G)$ is the materialization of G, *i.e.*, the set of all formulae $a \rightarrow x$ where $(a, x) \in G$.

The authors' papers [8] and [9, section 1] investigate these systems in detail – semantically, in terms of their explicit definitions, derivationally, in terms of the rules determining them, both separately and in relation to each other. We do not attempt to summarize the results here, but hope that the reader is tempted to follow further.

3 Why constrain?

As mentioned in section 1, all approaches to deontic logic must face the problem of dealing with contrary-to-duty norms. In general terms, we recall, the problem is: given a set of norms, how should we determine which obligations are operative in a situation that already violates some among them.

The following simple example is adapted from Prakken and Sergot [13].[1] Suppose we have the following two norms: *The cottage should not have a fence or a dog; if it has a dog it must have both a fence and a warning sign.*

In the usual deontic notation: $O(\neg(f \vee d)/t)$, $O(f \wedge w/d)$, where t stands for a tautology; in the notation of input/output logic: $(t, \neg(f \vee d))$, $(d, f \wedge w)$. Suppose further that we are in the situation that the cottage has a dog, thus violating the first norm. What are our current obligations?

Unrestricted input/output logic gives f: *the cottage has a fence* and w: *the cottage has a warning sign.* Less convincingly, because unhelpful if the presence of a dog is regarded as unalterable, it also gives $\neg d$: *the cottage does not have a*

[1] There are many examples in the literature. Most of them involve ingredients that, while perfectly natural in ordinary discourse, are extraneous to the essential problem and thus invite false analyses. These ingredients include defeasibility, causality, the passage of time, and the use of questionable rules such as CT and OR in deriving output. We have chosen a very simple example that avoids all those elements. There is one respect in which it could perhaps be further purified: under input d, the output is not only inconsistent with the input, but also itself inconsistent. This matter is discussed at the end of section 5.

dog. Even less convincingly, it gives $\neg f$: *the cottage does not have a fence*, which is the opposite of what we want.

These results hold even for simple-minded output, without reusability or disjunction of inputs. The only rules needed are SI and WO, as shown by the following derivation of $\neg f$.

$$\cfrac{\cfrac{(t, \neg(f \vee d))}{(t, \neg f)}\ \text{WO}}{(d, \neg f)}\ \text{SI}$$

A common reaction to examples such as these is to ask: why not just drop the rule SI of strengthening the input? In semantic terms, why not cut back the definition of simple-minded output from $Cn(G(Cn(A)))$ to $Cn(G(A))$, and in similar (but more complex) fashion with the others? Indeed, this is a possible option, and the strategy that we will describe below does have the effect of disallowing certain applications of SI. But simply to drop SI is, in the view of the authors, too heavy-handed. We need to know *why* SI is not always appropriate and, especially, *when* it remains justified.

4 A Strategy for Constraint: Maxfamilies and their Outfamilies

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 given norms.

Specifically, we look at the maximal subsets $G' \subseteq G$ such that$out(G', A)$ is consistent with input A. In [8], 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).[2]

To illustrate this, consider $G = \{(t, \neg(f \vee d)), (d, f \wedge w)\}$, 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 $Cn(f \wedge w)$.

[2] So defined, the outfamily is not in general the same as the family of all maximal values of $out(G', A)$ consistent with A, for G' ranging over subsets of G. Every maximal value of $out(G', A)$ is in the outfamily, but not always conversely. For certain of our output operations, the two families do coincide, but not for others.

This can be shown by simple examples, such as the Möbius strip of Makinson [6,7]. Put $G = \{(a, x), (x, y), (y, \neg a)\}$. Then, for $out = out_3$ or $out = out_4$, *maxfamily*(G, a) has three elements, namely the three two-element subsets of G. As a result, *outfamily*(G, a) also has three elements – $Cn(\emptyset)$, $Cn(x)$, and $Cn(\{x, y\})$. Of these, only the last is a maximal value of $out(G', A)$ consistent with A for G' ranging over subsets of G.

We add that in this example, not even $Cn(\{x, y\})$ is a maximal subset of $out(G, a)$ that is consistent with a, for clearly $Cn(\{x, y\}) \subset Cn(\{x, y, \neg a \vee z\}) \subset out(G, a)$. Care is thus needed to avoid confusing maxfamilies with related maximal sets.

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 (t, x), then for the degenerate input/output operation $out_2^+(G, a) = out_4^+(G, a) = Cn(m(G) \cup \{a\})$, the elements of *outfamily*(G, a) are just the maxichoice revisions of $m(G)$ by a, in the sense of Alchourrón, Gärdenfors and Makinson [3]. These coincide, in turn, with the extensions of the default system $(m(G), a, \emptyset)$ of Poole [12].

More surprisingly, there are close connections with the default logic of Reiter, falling a little short of identity. Read elements (a, x) of G as normal default rules $a; x/x$ in the sense of Reiter [14], and write *extfamily*(G, A) for the set of extensions of (G, A). Then, for reusable simple-minded throughput out_3^+, it can be shown that *extfamily*$(G, A) \subseteq$ *outfamily*(G, A) and indeed that *extfamily*(G, A) consists of precisely the maximal elements (under set inclusion) of *outfamily*(G, A).

These results and related ones are proven in Makinson and van der Torre [9]. But in accord with the motivation from the logic of norms, the main focus in that paper is on input/output logics *without* throughput. Two kinds of question are investigated in detail there.

4.1 The search for truth-functional reductions of the consistency constraint

From the point of view of computation, it is convenient to make consistency checks as simple as possible, and executable using no more than already existing programs. For this reason, it is of interest to ask: under what conditions is the consistency of A with $out(G, A)$ reducible to the consistency of A with the materialization $m(G)$ of G, *i.e.*, with the set of all formulae $a \to x$ where $(a, x) \in G$?

It is easy to check that the latter consistency implies the former for all seven of our input/output operations. It turns out that we have equivalence for just two of them (reusable basic with and without identity).

On the level of derivations, the question can take a rather different form, with different answers. Given a derivation of (a, x) with leaves L, under what conditions is the consistency of a with $out(L, a)$ equivalent to its consistency with $m(L)$? Curiously, this holds for a wider selection of our input/output operations – in fact, for all of them except basic output. Even more surprisingly, for some of the operations (those without OR), the same reduction also holds with respect to the set $h(L)$ of heads x, and the set $f(L)$ of fulfilments $a \wedge x$, of elements (a, x) of L.

From this result on derivations, we can go back and sharpen the semantic one. When G is a *minimal* set with $x \in out(G, a)$ then, for each of our input/output operations other than basic output, a is consistent with $out(G, a)$ iff it is consistent with $m(G)$ – and for the operations without OR, with $h(G)$, $f(G)$.

4.2 More severe applications of the consistency check

From a practical point of view, whenever we constrain an operation to avoid excess production, the question arises: how cautious (timid) or brave (foolhardy) do we want to be? For input/output operations, this issue arises in different ways on the semantic and derivational levels. On the semantic level, once we have formed an outfamily we may ask: should we intersect, join, or choose from its elements to obtain a unique restrained output? On the level of derivations, it is natural to ask: do we want to apply the consistency check only at the root of a derivation, or at every step within it?

The policy of checking only at the root corresponds to the option, on the semantic level, of forming the join of the outfamily; while the stricter policy of checking at every step is an essentially derivational requirement. But whichever of the two we choose, it is of interest to know under what conditions they coincide. In other words, given a derivation of (a, x) with leaves L such that a is consistent with $out(L, a)$, under what conditions does it follow that for every node (b, y) in the derivation, b is consistent with $out(L, b)$? It turns out that for certain of the seven input/output operations (again, those without the OR rule) this result holds. For operations with OR but without the rule CT, a rather subtler result may be obtained.

One lesson of these rather intricate investigations is that the behaviour of the consistency constraint depends very much on the choice of input/output operation; in particular, the presence of the rule OR destroys some properties. Another lesson is that questions can take different forms, with different answers, on the semantic and derivational levels. Thirdly, a detour through derivations can sometimes sharpen semantic results.

5 Doubts and Queries

The investigation of constrained output is a much more complex matter than that of unconstrained output. It is also more open to doubts and queries. We put the main ones on the table.

5.1 Dependence on the formulation of G

The outfamily construction, at least in its present form, depends heavily on the formulation of the generating set G. To illustrate this, we go back to the cottage example of Prakken and Sergot [13] considered in sections 3 and 4. Here $G = \{(t, \neg(f \vee d)), (d, f \wedge w)\}$, and we consider the contrary-to-duty input d. As we have seen, using simple-minded output, *maxfamily*(G, d) has unique element $\{(d, f \wedge w)\}$ and *outfamily*(G, d) has unique element $Cn(f \wedge w)$. But if we split the first element of G into $(t, \neg f), (t, \neg d)$ then we get a different result. The maxfamily has two elements $\{(t, \neg f)\}$, $\{(d, f \wedge w)\}$ and the outfamily has two elements $Cn(\neg f)$ and $Cn(f \wedge w)$. Is this dependence on formulation of G a virtue, or a vice?

5.2 Are we cutting too deeply?

This problem is related to the first one. In some cases, the outfamily construction cuts deeply, perhaps too much. Consider again the cottage example, but this time with just one rule $(t, \neg(f \vee d))$ in G. Consider the same contrary-to-duty input d. Then the maxfamily has the empty set as its unique element, and so the outfamily has $Cn(\emptyset)$ as its unique element. Is this cutting too deeply? Shouldn't $Cn(\neg f)$ be retained?

5.3 Should we pre-process G?

If we wish to cut less deeply, then a possible procedure might be to 'pre-process' G. In the last example, when we decompose the sole element $(t, \neg(f \vee d))$ of G into $(t, \neg f)$, $(t, \neg d)$ then $Cn(\neg f)$ becomes the unique element of outfamily in the contrary-to-duty situation d. In general, for each element (a, x) of G, we could rewrite the head x in conjunctive normal form $x_1 \wedge \ldots \wedge x_n$, and then split (a, x) into $(a, x_1), \ldots, (a, x_n)$. This manoeuvre certainly meets the particular example. But is it appropriate for other examples of the same form with different content? And does it suffice for more complex examples? It looks suspiciously like hacking.

5.4 Avoid inconsistency with what?

On our definition, $maxfamily(G, A)$ is the family of maximal subsets $G' \subseteq G$ such that $out(G', A)$ is consistent with input A. It may be suggested that this is too radical – so long as $out(G, A)$ is consistent we should apply it without constraint.

To illustrate this, take another variation on the cottage example. Put $G = \{(t, \neg(f \vee d)), (d, w)\}$. The second norm no longer requires a fence when there is a dog, only a warning sign. Consider again the contrary-to-duty input d. Now $out(G, d) = Cn(\{(\neg f, \neg d, w\})$ which is inconsistent with the input d, but itself perfectly consistent. Should we cut it at all? Perhaps 'yes' if the input d is considered as unalterably true, but 'no' if it is presented as true but changeable.

6 Conditional Permission from an Input/output Perspective

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.

Positive permission is more elusive. As a first approximation, one may say that something is positively permitted by a code iff the code explicitly presents it as such. But this leaves the central logical question unanswered. As well as the items that a code explicitly pronounces to be permitted, there are presumably others that in some sense follow from the explicit ones. The problem is to make it clear what kind of 'following' this is.

From the point of view of input/output logic, negative permission is straightforward to define: we simply put $(a, x) \in \mathit{negperm}(G)$ iff $(a, \neg x) \notin \mathit{out}(G)$, where *out* is any one of the four input/output operations that we have already discussed.

Because of its negative character, *negperm* fails the rule SI (strengthening the input). In other words, we don't have: $(a, x) \in \mathit{negperm}(G) \& a \in Cn(b) \Rightarrow (b, x) \in \mathit{negperm}(G)$. Indeed, it satisfies the opposite rule WI (weakening the input): $(a, x) \in \mathit{negperm}(G) \& b \in Cn(a) \Rightarrow (b, x) \in \mathit{negperm}(G)$. For if $(a, \neg x) \notin \mathit{out}(G)$ and $b \in Cn(a)$ then by SI for the underlying output operation, $(b, \neg x) \notin \mathit{out}(G)$ so $(b, x) \in \mathit{negperm}(G)$. This is a particular instance of a quite general pattern: whenever out satisfies a Horn rule (HR) then the corresponding *negperm* operation satisfies an 'inverse' Horn rule $(\mathrm{HR})^{-1}$.

How should we define positive permission for conditional norms? Let G, P be sets of ordered pairs of propositions, where G represents the explicitly given conditional obligations of a code and P its explicitly given conditional permissions. The operation of *forward positive permission* is defined by putting:

$(a, x) \in \mathit{forperm}(P, G)$ iff $(a, x) \in \mathit{out}(G \cup Q)$ for some singleton or empty $Q \subseteq P$

i.e. in the principal case that P is not itself empty,

$(a, x) \in \mathit{forperm}(P, G)$ iff $(a, x) \in \mathit{out}(G(c, z))$

for some pair $(c, z) \in P$. This 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.

On the other hand, the operation of *backward positive permission* is defined by setting:

$(a, x) \in \mathit{backperm}(P, G)$ iff $(c, \neg z) \notin \mathit{out}(G \cup \{(a, x)\})$ for some pair $(c, z) \in P$ with c consistent.

This 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 something that has been explicitly permitted. With this in mind, one could also speak of the operation as one of *prohibition immunity.*

What do these two notions mean in ordinary life? 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 has been explicitly permitted, then adding the prohibition is inadmissible under pain of incoherence, and the pair (a, x) is to that extent protected from prohibition.

Forperm and *backperm* are very different operations. Whereas *forperm* satisfies SI, *backperm* satisfies WI. Like negative permission, *backperm* satisfies the 'inverse' rule $(\mathrm{HR})^{-1}$ of any Horn rule (HR) satisfied by out; but *forperm* satisfies instead a 'subverse' rule $(\mathrm{HR})^{\downarrow}$.

Backperm may be characterized in a rather different way, using an idea of Makinson, [7]. Let us say that G is cross-coherent with P iff there is no $(c, z) \in P$ with c consistent, such that $(c, \neg z) \in \mathit{out}(G)$. Then it is easy to check that $(a, x) \in \mathit{backperm}(P, G)$ iff $(a, x) \in \mathit{negperm}(H)$ for every $H \supset G$ that is cross-coherent with P. From this it follows, in particular, that when G is cross-coherent with P then $\mathit{backperm}(P, G) \subseteq \mathit{negperm}(G)$. In this sense, we can say that under 'normal conditions' backward permission is a strengthened negative permission.

Further details of the behaviour of these operations may be found in Makinson and van der Torre [10].

7 Conclusions

Drawing together the threads of this paper, we emphasize the main points.

– Input/output logic seeks to extract the essential mathematical structure behind recent attempts to reconstruct deontic logic that avoid treating norms as if they had truth-values.
– Unconstrained input/output provides us with a simple and elegant construction with straightforward behaviour, but whose application to norms totally ignores the subtleties of contrary-to-duty obligations.
– On the other hand, output constrained using the outfamily strategy provides a way of dealing with contrary-to-duty obligations. Its behaviour is quite subtle, and depends considerably on the choice of background input/output operation, in particular on whether or not it authorizes the rule of disjunction of inputs.
– However, our definition of an outfamily has features that might be regarded as shortcomings. Its effect depends on the formulation of the generating set of norms; in some examples it gives what may be regarded as a wrong result unless some pre-processing as carried out on the generating set; and in some contexts the requirement of consistency of output with input may be too strong. These are delicate issues, and it remains possible that they have no unique solution definable in purely formal terms.

– Input/output operations also enable us to give a clear formal articulation of the well-known distinction between negative and positive permission. They also enable us, for the first time, to distinguish two very different kinds of positive permission, with quite different uses in practical life.

A topic of further research is the analysis of structured assemblies of input/output operations. Such structures, called logical input/output nets, or lions for short, are graphs, with the nodes labelled by pairs (G, out) where G is a normative code and out is an input/output operations (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.

References

1. Alchourrón, C., "Philosophical foundations of deontic logic and the logic of defeasible conditionals", in: Meyer, J. and Wieringa, R. (eds.), *Deontic Logic in Computer Science*, New York: Wiley, 1993, 43–84.
2. Alchourrón, C. and Bulygin, E., "The expressive conception of norms", in: Hilpinen, R. (ed.), *New Essays in Deontic Logic*, Dordrecht: Reidel, 1981, 95–124.
3. Alchourrón, C., Grdenfors, P. and Makinson, D., "On the logic of theory change: partial meet contraction and revision functions", *The Journal of Symbolic Logic*, **50**, 1985, 510–530.
4. Dubislav, W., "Zur Unbegrndbarkeit der Forderungstze", *Theoria*, **3**, 1937, 330–342.
5. Jørgensen, J., "Imperatives and logic", *Erkenntnis*, **7**, 1937-8, 288–296.
6. Makinson, D., "General Patterns in Nonmonotonic Reasoning", in: Gabbay, H. and Robinson (eds.), *Handbook of Logic in Artificial Intelligence and Logic Programming*, vol. 3, Oxford University Press, 1994, 35–110.
7. Makinson, D., "On a fundamental problem of deontic logic", in: McNamara, P. and Prakken, H. (eds.), *Norms, Logics and Information Systems. New Studies in Deontic Logic and Computer Science*, vol. 49 of *Frontiers in Artificial Intelligence and Applications*, Amsterdam: IOS Press, 1999, 29–53.
8. Makinson, D. and van der Torre, L., "Input/output logics", *J. Philosophical Logic*, **29**, 2000, 383–408.
9. Makinson, D. and van der Torre, L., "Constraints for input/output logics", *J. Philosophical Logic*, **30(2)**, 2001, 155–185.
10. Makinson, D. and van der Torre, L., "Permission from an input/output perspective", *J. Philosophical Logic*, **32(4)**, 2003, 391–416.
11. Makinson, D. and van der Torre, L., "What is Input/Output Logic?", in: *Foundations of the Formal Sciences II: Applications of Mathematical Logic in Philosophy and Linguistics*, vol. 17 of *Trends in Logic*, Kluwer, 2003.
12. Poole, D., "A logical framework for default reasoning", *Artificial Intelligence*, **36**, 1988, 27–47.
13. Prakken, H. and Sergot, M., "Contrary-to-duty obligations", *Studia Logica*, **57**, 1996, 91–115.
14. Reiter, R., "A logic for default reasoning", *Artificial Intelligence*, **13**, 1980, 81–132.
15. Stenius, E., "Principles of a logic of normative systems", *Acta Philosophica Fennica*, **16**, 1963, 247–260.