DAVID MAKINSON and LEENDERT VAN DER TORRE

PERMISSION FROM AN INPUT/OUTPUT PERSPECTIVE

Received 10 December 2002

ABSTRACT. Input/output logics are abstract structures designed to represent conditional obligations and goals. In this paper we use them to study conditional permission. This perspective provides a clear separation of the familiar notion of negative permission from the more elusive one of positive permission. Moreover, it reveals that there are at least two kinds of positive permission. Although indistinguishable in the unconditional case, they are quite different in conditional contexts. One of them, which we call static positive permission, guides the citizen and law enforcement authorities in the assessment of specific actions under current norms, and it behaves like a weakened obligation. Another, which we call dynamic positive permission, guides the legislator. It describes the limits on the prohibitions that may be introduced into a code, and under suitable conditions behaves like a strengthened negative permission.

KEY WORDS: conditional norms, deontic logic, input/output logics, permission

1. INTRODUCTION

In formal deontic logic, permission is studied less frequently than obligation. For a long time, it was naively assumed that it can simply be taken as a dual of obligation, just as possibility is the dual of necessity in modal logic. As time passed, more and more researchers realized how subtle and multi-faceted the concept is. Nevertheless, they continued focussing on obligation because there it is easier to make progress within existing paradigms. Consequently the understanding of permission is still in a less satisfactory state.

Nevertheless, in more philosophical discussions it is common to distinguish between two kinds of permission, negative and positive.[1] The former is straightforward to describe: something is permitted by a code iff it is not prohibited by that code. That is, understanding prohibition in the usual way, 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 a central logical question unanswered. As well as the items that a code explicitly pronounces to be permitted, there

Journal of Philosophical Logic **32:** 391–416, 2003.
© 2003 *Kluwer Academic Publishers. Printed in the Netherlands.*

are others that in some sense follow from the explicit ones. The problem is to clarify the inference from one to the other.

An informal example illustrates this problem, at the same time bringing out the need to distinguish different kinds of permission. Suppose that a normative code contains just one explicit obligation and one explicit permission, both conditional. The obligation tells us that filling in an annual income-tax form is required, on the condition of being in gainful employment. The permission tells us that voting in elections is permitted, on the condition of being at least 18 years of age. We ask some questions.

- Does it follow from our mini-code that voting is permitted on condition of being employed?

In one sense *yes*, for there is nothing in the code that forbids it. This is negative permission. In another sense *no*, for a person may be employed at the age of 17 and not covered by the explicit permission. This is one kind of positive permission. But in another sense, *yes* again. For if we were to forbid a person to vote on the condition of being employed we would be creating an incoherence in the code, in its application to people who are both employed and 18 or over. This is another kind of positive permission.

But is that second kind of positive permission any different from negative permission? Another question brings out the difference.

- Does it follow from our code that drinking alcohol is permitted on condition of being 18 or over?

As far as negative permission is concerned, the answer is again *yes*: there is nothing in the code that forbids it. But from the positive point of view, the answer is *no*, whichever of the two kinds of positive permission we have in mind. In the first place, the mini-code does not mention alcohol, and contains no explicit permission that would cover those 18 or over who drink it. In the second place, we *can* add to the code by prohibiting those 18 or over from drinking alcohol without rendering the code incoherent.

To deal with examples such as this we make use of input/output operations. These operations make use of classical consequence, but do so passing through a set of so-called generators representing an explicitly given normative code. They were introduced in (Makinson and van der Torre, 2000) to analyse conditional obligations; we now use them to throw light on permissions.

We begin by examining negative permission from an input/output perspective. We then show how there are at least two quite different kinds of positive permission in conditional contexts, which we call ‘static’ and ‘dynamic’ operations. The properties of each are studied in some detail. Roughly speaking, static positive permission inherits many properties from

the underlying input/output operation, and behaves like a diminished obligation. On the other hand, dynamic positive permission behaves like an amplified negative permission.

To keep the presentation to a reasonable length we must assume some familiarity with input/output operations, but to help the reader the basic concepts are set out in Appendix 1. For further background, see the overview (Makinson and van der Torre, 2003) or the detailed study (Makinson and van der Torre, 2000).[2]

2. Negative Conditional Permission

We begin by analysing negative conditional permission from the standpoint of input/output logic. Although its definition is trivial, its consequences are sometimes surprising and they provide a useful point of comparison with the positive operations to be defined later.

Recall again the intuitive idea: something is permitted by a code iff it is not forbidden by that code. In two little words: iff *nihil obstat*. For conditional norms, this becomes: a code permits something with respect to a condition iff it does not forbid it under that same condition, i.e. iff the code does not require the contrary under that condition.[3]

To express this concept in terms of input/output operations, let G be a set of ordered pairs (a, x) of Boolean propositions, representing a code of obligations. We put $(a, x) \in \mathit{negperm}(G)$ iff $(a, \neg x) \notin \mathit{out}(G)$. Here $\mathit{out} = \mathit{out}_i$ for $i \in \{1, 2, 3, 4\}$ is any one of the four input/output operations without throughput that are defined in Appendix 1 and studied in (Makinson and van der Torre, 2000).

Negative permission is not always implied by obligation. In other words, we do not always have $\mathit{out}(G) \subseteq \mathit{negperm}(G)$. This can fail in two distinct ways. It fails whenever G is *internally incoherent*, in the sense that there is a classically consistent proposition a with both $(a, x) \in \mathit{out}(G)$ and $(a, \neg x) \in \mathit{out}(G)$, for the latter tells us that $(a, x) \notin \mathit{negperm}(G)$. It can also sometimes fail for internally coherent codes, considering pairs of the form (f, x) where f is a contradiction. For example, let $G = \{(a, x), (\neg a, \neg x)\}$. This tells us that x should hold in the case a, and that $\neg x$ should hold in the case that $\neg a$, and the code is perfectly coherent, both by our definition above and under any intuitive point of view. But clearly by SI (Strengthening the Input – see Appendix 1), we have both $(f, x) \in \mathit{out}(G)$ and $(f, \neg x) \in \mathit{out}(G)$, the latter telling that $(f, x) \notin \mathit{negperm}(G)$.

The relationship between $\mathit{out}(G)$ and $\mathit{negperm}(G)$ may be expressed as follows. When A, B are sets of pairs, we say that A is *almost included* in B, and write $A \subseteq^{c} B$, iff whenever $(a, x) \in A$ and a is classically consistent

then $(a, x) \in B$. Then immediately we have: $out(G) \subseteq^c negperm(G)$ iff G is internally coherent in the sense defined above.

Whereas the input/output operations out_i for $i \in \{1, 2, 3, 4\}$ are all closure operations, *negperm* is not. It is easy to check that monotony in G is replaced by antitony and that idempotence fails outright. Inclusion does not hold in general, but since $G \subseteq out(G)$ and as just noted $out(G) \subseteq^c negperm(G)$ whenever G is internally coherent we do have $G \subseteq^c negperm(G)$ under that same condition.

Again, all of our input/output operations satisfy the rule SI (strengthening the input), i.e. the rule $(a, x) \in out(G)$ & $a \in Cn(b) \Rightarrow (b, x) \in out(G)$ where Cn, we recall, is classical consequence. On the other hand, negative permission does not.[4]

Indeed, one might imagine that the operation satisfies few if any Horn rules, but this is not the case. For example, for any output operation including out_1, the corresponding *negperm* operation satisfies the very opposite of SI:

WI (weakening the input): $(a, x) \in negperm(G)$ & $b \in Cn(a)$
$\Rightarrow (b, x) \in negperm(G)$.

For if $(a, \neg x) \notin out(G)$ and $b \in Cn(a)$ then by the rule SI (strengthening the input) for the underlying output operation, $(b, \neg x) \notin out(G)$.

Also, like the underlying input/output operation, *negperm* satisfies the following one-premise Horn rule:

WO (weakening the output): $(a, x) \in negperm(G)$ & $y \in Cn(x)$
$\Rightarrow (a, y) \in negperm(G)$.

For if $(a, \neg x) \notin out(G)$ and $y \in Cn(x)$ so that $\neg x \in Cn(\neg y)$ then by WO for *out*, $(a, \neg y) \notin out(G)$.

For quite different reasons, *negperm* satisfies the rule:

TRIV (trivialization): $(t, f) \in negperm(G) \Rightarrow (a, x) \in negperm(G)$.

Here t is any tautology and f any contradiction. For $(t, t) \in out(G)$, so $(t, f) \notin negperm(G)$, making TRIV vacuously true for *negperm*.

A common pattern underlies these three rules, and extends to multi-premise rules. Consider any Horn rule for output of the following form:

(HR): $(\alpha_i, \varphi_i) \in out(G) (0 \leq i \leq n)$ & $\theta_j \in Cn(\gamma_j) (0 \leq j \leq m)$
$\Rightarrow (\beta, \psi) \in out(G)$.

We call $(\alpha_i, \varphi_i) \in out(G)$ the *substantive* premises, and $\theta_j \in Cn(\gamma_j)$ the *auxiliary* ones. Each of the rules used in (Makinson and van der Torre,

2000) to characterise the four output operations out_i with $i \in \{1, 2, 3, 4\}$ and their extensions by throughput, is of this form, indeed with $n \leq 2$ and $m \leq 1$. Now consider the following rule for the corresponding negative permission operation:

$$(\mathrm{HR})^{-1}\colon\ (\alpha_i, \varphi_i) \in out(G)(i < n)\ \&\ (\beta, \neg\psi) \in negperm(G)\ \&\ \theta_j \in Cn(\gamma_j)(j \leq m) \Rightarrow (\alpha_n, \neg\varphi_n) \in negperm(G).$$

In other words, the conclusion of (HR) is swapped with one of the substantive premises, negating both their heads in the process and rewriting their *out* as *negperm*, while all the other premises are left unchanged. In the limiting case that the output rule has no substantive premises, i.e. when $n = 0$, $(\mathrm{HR})^{-1}$ is understood to have an arbitrary pair as conclusion.

We call $(\mathrm{HR})^{-1}$ the *inverse* of (HR). Then immediately from the definition of *negperm* we have:

OBSERVATION 1. *Let out be any output operation. If out satisfies a rule of the form* (HR)*, then the corresponding negperm operation satisfies* $(\mathrm{HR})^{-1}$.

As particular cases with $n = 1$ (and eliminating surplus negation signs) WI is the inverse of SI, and WO is its own inverse. With $n = 0$, TRIV is the inverse of the rule TAUT: $\varnothing \Rightarrow (t, t) \in out(G)$ that holds of all our four underlying input/output operations.

When $n \geq 2$, i.e. when (HR) has more than one substantive premise, $(\mathrm{HR})^{-1}$ will be a Horn rule of a rather unusual kind. It will be 'mixed', in the sense that one premise is of the form $(\beta, \neg\psi) \in negperm(G)$ while others are of the form $(\alpha_i, \varphi_i) \in out(G)$. Moreover, in this case there will be more than one inverse; in fact n of them, according to which of the n substantive premises (α_i, φ_i) of the initial rule is selected for swapping. If the rule for output is symmetric, as are AND, OR, then its inverses will be merely notational variants of each other. But we get quite different inverses when the output rule is asymmetric, as in the case of cumulative transitivity CT: $(a, x) \in out(G)\ \&\ (a \wedge x, y) \in out(G) \Rightarrow (a, y) \in out(G)$.

Spelling these remarks out in detail: in the case of AND, the initial rule for output is:

$$(a, x) \in out(G)\ \&\ (a, y) \in out(G) \Rightarrow (a, x \wedge y) \in out(G)$$

so that its inverse is:

$$(a, x) \in out(G)\ \&\ (a, \neg(x \wedge y)) \in negperm(G) \Rightarrow (a, \neg y) \in negperm(G).$$

For OR the initial rule is:

$$(a, x) \in out(G) \;\&\; (b, x) \in out(G) \Rightarrow (a \vee b, x) \in out(G)$$

with inverse:

$$(a, x) \in out(G) \;\&\; (a \vee b, \neg x) \in negperm(G) \Rightarrow (b, \neg x) \in negperm(G).$$

For CT the initial rule is asymmetric:

$$(a, x) \in out(G) \;\&\; (a \wedge x, y) \in out(G) \Rightarrow (a, y) \in out(G)$$

and its two inverses are:

$$(a, x) \in out(G) \;\&\; (a, \neg y) \in negperm(G) \Rightarrow (a \wedge x, \neg y) \in negperm(G),$$

$$(a \wedge x, y) \in out(G) \;\&\; (a, \neg y) \in negperm(G) \Rightarrow (a, \neg x) \in negperm(G).$$

On the other hand, it is not clear how these Horn rules could lead to a characterization of *negperm* as the closure of some basis under the rules, for it is not clear what the basis could be.[5] We will see that with positive permission, the situation is quite different in this respect.

3. Positive Permission: A Tale of Two Definitions

How can we define positive permission for conditional norms? A natural idea is to begin by considering the unconditional case and see whether it offers an answer that can be adapted to the conditional one.

When all norms under consideration are unconditional, we may apply classical consequence in a quite straightforward way. Suppose that our code consists of a set A of Boolean propositions, representing the items that are explicitly obligated, and another set Z of Boolean propositions representing those explicitly permitted. We may take a proposition x to be positively permitted by the code iff there is a $z \in Z$ with $x \in Cn(A \cup \{z\})$, where Cn is classical consequence. In this definition, the elements of A may be used jointly, while the elements of Z can only be used one by one, so that even when x, x' are both permitted their conjunction $x \wedge x'$ need not be so.[6]

This definition can be put in another form, which is trivially equivalent but will provide a quite different perspective when we consider conditional

norms. A proposition x is positively permitted by the code iff there is a $z \in Z$ with $\neg z \in Cn(A \cup \{\neg x\})$. The two formulations are equivalent by contraposition for classical consequence.

But when we pass to conditional norms, then the situation becomes less straightforward, as classical consequence cannot be applied directly. It is not meaningful to talk of the *classical* consequences of a conditional obligation with condition a and obligated x, unless that is identified with a formula of classical logic, say the material implication $a \rightarrow x$. But, as is well known, this identification gives highly counterintuitive results, notably validation of contraposition and the identity principle for conditional norms. Of course, one could try representing conditional obligation by means of a non-classical connective and applying some kind of non-classical logic. Indeed, that is the standard approach, but in this paper we follow another one, using input/output operations.

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. For example, P could include all those items that the code explicitly declares to be 'rights'. The new element P is a vital component of all concepts that we now introduce.

For *static positive permission*, the idea is to treat (a, x) as permitted iff there is some explicitly given permission (c, z) such that when we join it with the obligations in G and apply the output operation to the union, then we get (a, x). To be precise, in the principal case that P is non-empty we put:

$$(a, x) \in statperm(P, G) \quad \text{iff} \quad (a, x) \in out(G \cup \{(c, z)\})$$
$$\text{for some pair } (c, z) \in P.$$

In the limiting case that $P = \varnothing$, we simply set $(a, x) \in statperm(P, G)$ iff $(a, x) \in out(G)$.

Of course, these two cases could be expressed as one, by saying:

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

or equivalently, given the properties of the input/output operation:

$$(a, x) \in statperm(P, G) \quad \text{iff} \quad (a, x) \in out(G \cup \{(c, z)\})$$
$$\text{for some pair}(c, z) \in P \cup \{(t, t)\}.$$

Static permissions are thus treated like weak obligations, the basic difference being that while the latter may be used jointly, the former may only be applied one by one.

The operation of *dynamic positive permission* is considerably more complex. It is based on an idea of Alchourrón, and was given a crude formulation in (Makinson, 1999). Whereas the definition of static permission corresponds to the first of the two formulations for the unconditional case, given at the beginning of this section, the definition of dynamic permission corresponds to the second. But they are no longer equivalent, essentially because unlike plain classical consequence, input/output operations do not in general satisfy contraposition.

The idea is to see (a, x) as 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 explicitly permitted. The precise definition makes use of *statperm* as well as *out*:

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

How far do these two concepts correspond to what we do in ordinary life? Static permission seems to answer to the needs of the citizen, who needs to anticipate the deontic status of his actions. He needs, first of all, to assure himself that the action that he is entertaining is not forbidden, in other words, that it is negatively permitted. But given the practical difficulties of establishing a negative fact of this kind, which are multiplied by any ambiguities or vagueness in the code of obligations, he would also be glad to learn that his action is 'covered' by some explicit permission. Once the action is performed, the same questions need to be asked by authorities if they are called upon to assess it.

On the other hand, dynamic permission corresponds to the needs of the legislator, who needs to anticipate the effect of *changing* an existing corpus of norms by adding a prohibition. If prohibiting x in condition a would commit us to forbidding something that has been positively permitted in a certain realizable situation, then adding the prohibition would give rise to a certain kind of incoherence.

It is important to be clear about the kind of coherence at issue here, and not to confuse it with the internal coherence of $\mathit{out}(G)$ taken alone. We recall from Section 2 that G is called *internally coherent* iff there is no pair (c, z), with c classically consistent and both (c, z), $(c, \neg z) \in \mathit{out}(G)$. When dealing with dynamic permission, on the other hand, we need to consider coherence between G and P. We say that G is *cross-coherent* with P iff there is no pair (c, z), with c classically consistent and $(c, \neg z) \in \mathit{out}(G)$ while $(c, z) \in \mathit{statperm}(P, G)$. The definition of *dynperm* may

thus be equivalently expressed as follows:[7]

$$(a, x) \in \mathit{dynperm}(P, G) \quad \text{iff} \quad G \cup \{(a, \neg x)\} \text{ is not cross-coherent with } P.$$

EXAMPLE. To illustrate the concepts of negative, static and dynamic permission, we take the same example as in Section 1 and work it out formally.

Let $G = \{(\mathit{work}, \mathit{tax})\}$ and $P = \{(18, \mathit{vote})\}$, where $\mathit{work} = \mathit{John}$ is engaged in gainful employment, $\mathit{tax} = \mathit{John}$ fills in an annual income-tax form, $18 = \mathit{John}$ is at least 18 years old, $\mathit{votes} = \mathit{John}$ votes in elections. Put also $\mathit{male} = \mathit{John}$ is male, $\mathit{drink} = \mathit{John}$ drinks alcohol. Then we have the following pattern, irrespective of the choice of our background input/output operation out_i for $i \in \{1, 2, 3, 4\}$.

Pair	$\mathit{out}(G)$	$\mathit{statperm}(P, G)$	$\mathit{dynperm}(P, G)$	$\mathit{negperm}(P, G)$
$(\mathit{work} \wedge \mathit{male}, \mathit{tax})$	yes	yes	yes	yes
$(\mathit{work} \wedge 18, \mathit{vote})$	no	yes	yes	yes
$(\mathit{work}, \mathit{vote})$	no	no	yes	yes
$(18, \mathit{drink})$	no	no	no	yes
$(\mathit{work} \wedge \mathit{male}, \neg\mathit{tax})$	no	no	no	no

Some explanations may help clarify the individual entries.

- $(\mathit{work} \wedge \mathit{male}, \mathit{tax})$ is in $\mathit{out}(G)$ because $(\mathit{work}, \mathit{tax})$ is in G and *out* satisfies SI.
- $(\mathit{work} \wedge 18, \mathit{vote})$ is not in $\mathit{out}(G)$. But it is in $\mathit{statperm}(P, G)$ because $\mathit{statperm}(P, G) = \mathit{out}(\{(\mathit{work}, \mathit{tax}), (18, \mathit{vote})\})$ and *out* satisfies SI, which can be applied to the second element.
- On the other hand, $(\mathit{work}, \mathit{vote})$ is not in $\mathit{out}(\{(\mathit{work}, \mathit{tax}), (18, \mathit{vote})\})$ and so is not in $\mathit{statperm}(P, G)$. It is however in $\mathit{dynperm}(P, G)$, because when we add its 'opposite' $(\mathit{work}, \neg\mathit{vote})$ to G we enter into conflict with the existing static permissions. To be precise, $(\mathit{work} \wedge 18, \neg\mathit{vote}) \in \mathit{out}(G \cup \{(\mathit{work}, \neg\mathit{vote})\}$ while as we have just seen $(\mathit{work} \wedge 18, \mathit{vote}) \in \mathit{statperm}(P, G)$.
- Next, $(18, \mathit{drink})$ is not in $\mathit{dynperm}(P, G)$, because we can add its 'opposite' $(18, \neg\mathit{drink})$ to G without conflict with the existing static permissions. In other words, $\mathit{out}(\{(\mathit{work}, \mathit{tax}), (18, \neg\mathit{drink})\})$ does not contain any pair conflicting with one in $\mathit{statperm}(P, G) = \mathit{out}(\{(\mathit{work}, \mathit{tax}), (18, \mathit{vote})\})$. However $(18, \mathit{drink})$ is in $\mathit{negperm}(G)$ because $(18, \neg\mathit{drink}) \notin \mathit{out}(G)$.

- Finally, $(work \wedge male, \neg tax)$ is not even in $negperm(G)$ because, as we have noted, $(work \wedge male, tax)$ is in $out(G)$.

This analysis is in full accord with the intuitive assessment of the example made at the end of Section 1. We note that in this example we have a chain of inclusions $out(G) \subseteq statperm(P, G) \subseteq dynperm(P, G) \subseteq negperm(P, G)$, indeed with all inclusions proper.

In what follows we study the behaviour of static and dynamic permission in detail, comparing them to each other and to negative permission. For instance, we show that the inclusions just noted for the above example hold or 'almost hold', either in general or under suitable conditions. We also show how static and dynamic permission contrast in many respects. While, as we have remarked, the former resembles a diminished obligation, the latter is, for certain codes, a strengthened negative permission. Sections 4 and 5 focus on the static operation and Section 6 on the dynamic one.[8]

4. Properties of Static Positive Permission

It is immediate from the definition of static permission that $out(G) \subseteq statperm(P, G)$ no matter what the value of P. For *out* is monotone in G and thus when $(a, x) \in out(G)$ then $(a, x) \in out(G \cup \{(c, z)\})$ for any (c, z).

Its relations with negative permission are more complex. On the one hand, simple examples show that neither $statperm(P, G)$ nor $negperm(G)$ is always included in the other.[9] However, we do have a connection. It is immediate from the definitions that $statperm(P, G) \subseteq^{c} negperm(G)$ iff G is cross-coherent with P. Here cross-coherence is as defined in Section 3, and the 'almost inclusion' $\subseteq^{c}$ is as defined in Section 2.

Turning now to the properties of *statperm* itself, it is easy to verify that since $out(G)$ is a closure operation, $statperm(P, G)$ is a closure operation in its argument P. That is, $P \subseteq statperm(P, G) = statperm(statperm(P, G), G)$, and $P \subseteq Q$ implies $statperm(P, G) \subseteq statperm(Q, G)$.

Statperm also satisfies inclusion and monotony in its auxiliary argument G, i.e. $G \subseteq statperm(P, G)$, and $G \subseteq H$ implies $statperm(P, G) \subseteq statperm(P, H)$. But it is not a closure operation in the argument G, since it does not satisfy idempotence in that argument.[10]

Statperm is like *out* (and unlike *negperm*) in that it satisfies SI. Indeed, it inherits this property directly from *out*. For suppose $(a, x) \in statperm(P, G)$. Then $(a, x) \in out(G \cup Q)$ for some singleton or empty

$Q \subseteq P$, so using SI for *out* we have $(a \wedge b, x) \in out(G \cup Q)$ and thus $(a \wedge b, x) \in statperm(P, G)$.

This is an example of a very general pattern. Satisfaction of a Horn rule passes not to the inverse rule (as it does for *negperm*), but to what we may call the *subverse* rule. To be precise, consider again any Horn rule for output, of the form:

(HR): $(\alpha_i, \varphi_i) \in out(G)(i \leq n)$ & $\theta_j \in Cn(\gamma_j)(j \leq m)$
$\Rightarrow (\beta, \psi) \in out(G)$.

We define its *subverse* to be the rule:

(HR)$^{\downarrow}$: $(\alpha_i, \varphi_i) \in out(G)(i < n)$ & $(\alpha_n, \varphi_n) \in statperm(P, G)$
& $\theta_j \in Cn(\gamma_j)(j \leq m) \Rightarrow (\beta, \psi) \in statperm(P, G)$.

In other words, the subverse rule is obtained by downgrading to permission status one of the substantive premises and also the conclusion of the rule. In the limiting case that there are no substantive premises, the conclusion alone is downgraded. As before, when $n \geq 2$ strictly speaking there are n subverses, depending on which of the n substantive premises of the initial rule is selected for downgrading.

OBSERVATION 2. *Let out be any output operation. If out satisfies a rule of the form* (HR)*, then the corresponding statperm operation satisfies the subverse(s)* (HR)$^{\downarrow}$.

Proof. In the limiting case that $P = \varnothing$, $statperm(P, G) = out(G)$ and so the two rules are the same. For the principal case, suppose that *statperm* fails the subverse rule for the values P, G. Then there is a $(c, z) \in P$ with $(\alpha_n, \varphi_n) \in out(G \cup \{(c, z)\})$, while $(\alpha_i, \varphi_i) \in out(G) \subseteq out(G \cup \{(c, z)\})$ for all $i < n$, and $\theta_j \in Cn(\gamma_j)$ for all $j \leq m$, but $(\beta, \psi) \notin out(G \cup \{(c, z)\})$. Hence *out* fails the initial rule for the value $G' = G \cup \{(c, z)\}$. □

In the case that $n = m = 1$, the observation tells us that if *out* satisfies the rule:

$$(\alpha, \varphi) \in out(G) \;\&\; \theta \in Cn(\gamma) \Rightarrow (\beta, \psi) \in out(G)$$

then *statperm* satisfies the rule:

$$(\alpha, \varphi) \in statperm(P, G) \;\&\; \theta \in Cn(\gamma)$$
$$\Rightarrow (\beta, \psi) \in statperm(P, G),$$

i.e. satisfies the same rule with *statperm* written in place of *out*. This covers the case of SI mentioned above, as well as of WO. In the case that $n = 0$,

the observation implies that if *out* satisfies the rule TAUT, then so does *statperm*.

On the other hand, it does not seem that there are any non-trivial Horn rules with two or more substantive premises, i.e. in which $n \geq 2$, satisfied by both *out* and *statperm*. The reason is apparent from the example of the rule AND, which fails for *statperm*. Suppose we have $(a, x) \in out(G \cup \{(c, z)\})$ and $(a, y) \in out(G \cup \{(c', z')\})$ for some pairs $(c, z), (c', z') \in P$, so that $(a, x \wedge y) \in out(G \cup \{(c, z), (c', z')\})$. It does not follow that there is a single pair $(d, w) \in P$ with $(a, x \wedge y) \in out(G \cup \{(d, w)\})$. Similar arguments show the same for OR and CT.

This feature raises an interesting issue. On the one hand, the failure of AND corresponds to intuition: two actions may separately be permitted under a common condition without being jointly so. On the other hand, the failure of OR appears to be contrary to intuition, if we are working with an underlying input/output operation *out* that is itself assumed to satisfy OR. If x is permitted under condition a, and also under condition b, then why not under condition $a \vee b$?

For this reason, it may be preferable to strengthen the definition of *statperm* in its principal case, to ensure satisfaction of OR. This can be done by building into the definition of positive permission the same device that is used when defining basic output – intersect maximal supersets.

Recall that the principal case of our definition of static positive permission was:

$$(a, x) \in statperm(P, G) \quad \text{iff} \quad (a, x) \in out(G \cup \{(c, z)\}) \text{ for some pair } (c, z) \in P.$$

If we want *statperm* to satisfy OR, we should reformulate this. We know from (Makinson and van der Torre, 2000) or Appendix 1 of this paper that input/output operations are well-defined for pairs (A, x) where A is a set of propositions, as well as for pairs (a, x) where a is an individual proposition. Put:

$$(a, x) \in statperm_{\vee}(P, G) \quad \text{iff} \quad \text{for every complete set } V \text{ with } a \in V \text{ there is some pair } (c, z) \in P \text{ such that } (V, x) \in out(G \cup (c, z)).$$

This operation satisfies OR. For suppose that $(a \vee b, x) \notin statperm_{\vee}(P, G)$. Then there is a complete set V with $a \vee b \in V$ such that for all pairs $(c, z) \in P$ we have $(V, x) \notin out(G \cup (c, z))$. But since V is complete, either $a \in V$ or $b \in V$, so either $(a, x) \notin statperm_{\vee}(P, G)$ or $(b, x) \notin statperm_{\vee}(P, G)$.

Thus we have two versions of the corresponding static positive permission operation, *statperm* failing OR and $statperm_{\vee}$ satisfying it. When the underlying input/output operation itself satisfies OR, it may be preferable to select the latter.

The failure of CT for *statperm* is more difficult to assess in intuitive terms. We suggest that it may deserve to fail, because its introduction can transform a code that would otherwise be cross-coherent (in the sense defined at the end of Section 3) into one that is cross-incoherent. Consider the example where $P = \{(a, x), (a \wedge x, y)\}$ and $G = \{(a \wedge \neg x, \neg y)\}$. Intuitively, this code appears to be coherent, and with *statperm* defined above, it is cross-coherent. But if we amplify the definition of *statperm* to ensure satisfaction of CT, we get $(a, y) \in statperm(P, G)$ and so by SI, $(a \wedge \neg x, y) \in statperm(P, G)$, which is cross-incoherent with $(a \wedge \neg x, \neg y) \in out(G)$.

Admittedly, this is not a very conclusive argument against CT, and the situation deserves further analysis. In general terms we can thus say that the notion of *statperm* is not quite as single-valued as first appeared. As well as our basic definition, one can consider a variant satisfying OR, and may possibly wish to seek one satisfying CT.

5. Axiomatizing Static Positive Permission with the Subverse Rules

Observation 2 suggests a question. Given an output operation characterized by a set of Horn rules, do the subverses of those rules suffice to characterize the corresponding operation of static permission, taking P and G as bases for their application?

It will be convenient to follow the convention that in a tree serving as a derivation, only the premises that we called substantive (Section 2) are attached to nodes, while auxiliary premises $\theta \in Cn(\gamma)$ are attached to transitions. Thus, for example, a derivation of $(a, x \vee y)$ from (a, x) and $x \vee y \in Cn(x)$ by a single application of WO will have only one leaf node, labelled by the substantive premise (a, x), and a root node, decorated by $(a, x \vee y)$. The auxiliary premise $x \vee y \in Cn(x)$ of the rule will not be attached to a second leaf but to the transition.

With this convention in mind, and given the definition of the subverse of a Horn rule for output, the question above may be reformulated as follows. Is it the case that whenever $(a, x) \in out(G \cup Q)$ where Q is a singleton or empty subset of P, there is a derivation of (a, x) from $G \cup Q$ using the derivation rules characterizing *out*, in which we label each node with $out(G)$ or $perm(P, G)$ so that the following holds: (1) for every leaf, if it

carries a pair in $Q \backslash G$ then it is labelled *statperm*(P, G), (2) for every non-leaf, if it has a parent labelled *statperm*(P, G) then it is also so labelled, (3) no node has more than one parent labelled *statperm*(P, G)?

Clearly, when Q is empty, the three conditions are always satisfied. But what when Q is a singleton? The answer is negative, at least in the case of the output operation out_3 and the rules TAUT, SI, WO, AND, CT characterizing it (Makinson and van der Torre, 2000). For a counterexample, put $P = \{(a, x))\}$ and $G = \{(a \wedge x, y)\}$. On the one hand we have $(a, x \wedge y) \in out_3(G \cup \{(a, x)\})$, as witnessed by the following derivation:

$$\dfrac{\dfrac{(a, x) \qquad (a \wedge x, y)}{(a, y)}\,\mathrm{CT} \qquad (a, x)}{(a, x \wedge y)}\,\mathrm{AND}$$

On the other hand, if we try to effect a labelling of the above kind, we run into a difficulty arising from the fact that (a, x) is used twice. By condition (1), the two leaf nodes carrying (a, x) must both be labeled *statperm*(P, G), so by (2) the node carrying (a, y) must get the same label, so the node carrying $(a, x \wedge y)$ violates condition (3).

To be sure, this does not show that there is no *other* derivation using the same rules, of the same root from the same leaves, that does satisfy the three conditions, but it appears most unlikely.

It is easy to see that a derivation of (a, x) from $G \cup \{(c, z)\}$, where $(c, z) \notin G$, satisfies the three conditions iff at most one leaf node carries the pair (c, z). Our question about characterization by subverse rules may thus be put as follows. Under what conditions does a set of rules for an input/output operation satisfy the *non-repetition property*, that whenever $(a, x) \in out(G \cup \{(c, z)\})$ then there is a derivation of (a, x) from $G \cup \{(c, z)\}$, using those rules, such that (c, z) is attached to at most one leaf node? The example above shows that the non-repetition property does not always hold; the observation below shows that it very often does.

OBSERVATION 3. *The non-repetition property holds for:*

(a) out_1 *with its usual rules* TAUT, SI, WO, AND,
(b) out_2 *with its usual rules, i.e. the above plus* OR,
(c) out_3 *with the rules* TAUT, SI, WO, CTA.

Here CTA is the rule $(a, x) \in out(G)$ & $(a \wedge x, y) \in out(G) \Rightarrow (a, x \wedge y) \in out(G)$. This is not the usual set of rules for out_3, which consists of TAUT, SI, AND,WO plus CT: $(a, x) \in out(G)$ & $(a \wedge x, y) \in$

$out(G) \Rightarrow (a, y) \in out(G)$. The rule CTA in effect combines CT with AND into a single rule. The observation does not appear to hold for out_3 under its usual set of rules. In Appendix 2 we give proofs for out_1, out_2, out_3 (under the rules specified) and also a conjectured counterexample for out_4 (under the rules here used for out_3 plus OR).

From Observation 3 we thus have:

COROLLARY 4. *For each of the rule-sets mentioned in Observation* 3*, the subverse set suffices to characterize the corresponding static permission operation.*

6. Properties of Dynamic Positive Permission

We recall from Section 3 the definition of dynamic positive permission:

$$(a, x) \in dynperm(P, G) \quad \text{iff} \quad (c, \neg z) \in out(G \cup \{(a, \neg x)\})$$
$$\text{for some pair } (c, z) \in statperm(P, G) \text{ where } c \text{ is consistent.}$$

Immediately from the definition, $statperm(P, G) \subseteq^c dynperm(P, G)$, where the relation $\subseteq^c$ of 'almost inclusion' is as defined in Section 2. Since as already noted, $out(G) \subseteq statperm(P, G)$ we thus also have $out(G) \subseteq^c dynperm(P, G)$.

The *dynperm* operation is monotone in its argument P. The verification is immediate from the definition, using the monotony of *out* and the monotony of *statperm* in its argument P, noted in Section 4. *Dynperm* almost satisfies inclusion: we have $P \subseteq^c dynperm(P, G)$ but not full inclusion. Idempotence in P fails.[11] Thus, unlike *statperm*, it is not a closure operation in its argument P.

It is also a very different operation in its interaction with classical connectives. Whereas, as we have seen, *statperm* satisfies SI, *dynperm* (like *negperm*) satisfies WI. For suppose $(a, x) \in dynperm(P, G)$. Then $(c, \neg z) \in out(G \cup \{(a, \neg x)\})$ for some pair $(c, z) \in statperm(P, G)$ with c consistent. But by SI for *out*, $(a, \neg x) \in out(a \vee b, \neg x) \subseteq out(G \cup \{(a \vee b, \neg x)\})$ so since *out* is a closure operation, $(c, \neg z) \in out(G \cup \{(a \vee b, \neg x)\})$ and thus $(a \vee b, x) \in dynperm(P, G)$.

Thus in some respects, *dynperm* is surprisingly similar to negative permission. Its definition can be reformulated in a way that brings this out. Recall from Section 3 the definition of cross-coherence: we say that G is *cross-coherent* with P iff there is no classically consistent proposition c with both $(c, \neg z) \in out(G)$ and $(c, z) \in statperm(P, G)$. Then, refining an idea of (Makinson, 1999), we have the following characterization.

OBSERVATION 5. *The following three conditions are equivalent:*

(1) $(a, x) \in dynperm(P, G)$.
(2) $(G \cup \{(a, \neg x)\}$ *is not cross-coherent with* P.
(3) $(a, x) \in negperm(H)$ *for every* $H \supseteq G$ *that is cross-coherent with* P.

Proof. The equivalence of (1) and (2) is immediate from the definitions and was noted in Section 3 (just before the example). To complete the verification we check (1) $\Rightarrow$ (3) and (3) $\Rightarrow$ (2).

For (1) $\Rightarrow$ (3), suppose $(a, x) \in dynperm(P, G)$. Then there is a pair $(c, z) \in statperm(P, G)$ with c consistent and $(c, \neg z) \in out(G \cup \{(a, \neg x)\})$. Let $H \supseteq G$ be cross-coherent with P. Then $(c, \neg z) \notin out(H)$, so since *out* is a closure operation, $(a, \neg x) \notin out(H)$, i.e. $(a, x) \in negperm(H)$.

For (3) $\Rightarrow$ (2), suppose $(a, x) \in negperm(H)$ for every $H \supseteq G$ that is cross-coherent with P. Trivially $(a, \neg x) \in out(G \cup \{(a, \neg x)\})$ so $(a, x) \notin negperm(G \cup \{(a, \neg x)\}$, so putting $H = G \cup \{(a, \neg x)\}$ we have that H is not cross-coherent with P. □

COROLLARY 6. $Dynperm(P, G) \subseteq negperm(G)$ *iff* G *is cross-coherent with* P.

Proof. For right to left: suppose G is cross-coherent with P. Apply (1) $\Rightarrow$ (3) of Observation 5, putting $H = G$.

For left to right: suppose G is not cross-coherent with P. Then by definition, there is a $(c, z) \in statperm(P, G)$ with c consistent and $(c, \neg z) \in out(G)$. But we have already noted at the beginning of this section that $statperm(P, G) \subseteq^{c} dynperm(P, G)$, so $(c, z) \in dynperm(P, G)$ while $(c, \neg z) \in out(G)$ tells us that $(c, z) \notin negperm(G)$ and we are done. □

Thus, in a cross-coherent code, dynamic permission is a strengthened negative permission. Moreover, it behaves like negative permission as far as Horn rules are concerned. We have the following:

OBSERVATION 7. *Let out be any output operation. Every Horn rule of the kind* $(\mathrm{HR})^{-1}$ *satisfied by the corresponding negperm operation is also satisfied by dynperm.*

Proof. Consider any such rule, formulated for *dynperm*:

$$(\alpha_i, \varphi_i) \in out(G)(i < n) \;\&\; (\beta, \neg\psi) \in dynperm(P, G)$$
$$\;\&\; \theta_j \in Cn(\gamma_j)(j \leq m) \Rightarrow (\alpha_n, \neg\varphi_n) \in dynperm(P, G).$$

Suppose that the rule fails at the values P, G. We want to show that the corresponding rule for *negperm* fails at suitable values.

Since $(\beta, \neg\psi) \in dynperm(P, G)$, there is a pair $(c, z) \in statperm(P, G)$ with c consistent and $(c, \neg z) \in out(G \cup \{(\beta, \psi)\})$. On the other hand, since $(\alpha_n, \neg\varphi_n) \notin dynperm(P, G)$, we have $(c, \neg z) \notin out(G \cup \{(\alpha_n, \varphi_n)\})$. Put $H = G \cup \{(\alpha_n, \varphi_n)\}$. To show that the rule fails for *negperm* at the value H, it suffices to show that $(\alpha_n, \neg\varphi_n) \notin negperm(H)$ while $(\beta, \neg\psi) \in negperm(H)$. The former is immediate since $(\alpha_n, \varphi_n) \in out(H)$ by the definition of H. For the latter, suppose that $(\beta, \neg\psi) \notin negperm(H)$. Then $(\beta, \psi) \in out(H) = out(G \cup \{(\alpha_n, \varphi_n)\})$ so since *out* is a closure operation, $(c, \neg z) \in out(G \cup \{(\alpha_n, \varphi_n)\})$ giving us a contradiction. □

Thus *dynperm* has all the Horn properties of the form $(\text{HR})^{-1}$ that are possessed by *negperm*. At the same time it is a much more restricted operation, as shown by Corollary 6 and the example at the end of Section 3. It differs from *negperm* in certain formal properties in which G is allowed to vary; in particular, it is monotonic in G whereas *negperm* is antitonic in G.

This has implications for the way in which the logic of conditional permission is presented. The usual presentations leave unmentioned the sets G, P of explicit obligations and permissions. It then becomes very difficult to distinguish negative from dynamic permission on the basis of the Horn rules that they satisfy. They agree on those in which G is held fixed, and differ only on those in which G is allowed to vary, and if G is not represented then the effects of its variation become invisible. An adequate understanding of the different kinds of permission can only be effected if the sets of explicit obligations and permissions are also made explicit in the analysis.

Combining Observations 1 and 7 we immediately obtain the following:

OBSERVATION 8. *Let out be any output operation. If out satisfies a rule of the form* (HR)*, then the corresponding dynperm operation satisfies the inverse*(*s*) $(\text{HR})^{-1}$.

Thus in particular, for any output operation the corresponding *dynperm* operation satisfies the rules TRIV, WI, WO and the inverse of AND. If *out* satisfies OR or CT then the *dynperm* satisfies their respective inverses, written out in Section 2.

The results that we established in Section 5 on the characterization of static positive permission using only subverse rules, carry over *mutatis mutandis* to the problem of axiomatizing dynamic positive permission using only inverse rules. In particular, the problem reduces to the same one of the existence of a suitable labelling, and thus again to the presence of the non-repetition property. Thus, by parallel arguments we obtain:

COROLLARY 9. *For each of the rule-sets mentioned in Observation* 3*, the inverse set suffices to characterize the corresponding dynamic permission operation.*

7. SUMMARY AND CHARTS

We have shown how input/output operations provide a useful framework for the formal analysis of different kinds of permission. They permit a clear separation of the familiar notion of negative permission from the more elusive one of positive permission. Moreover, they reveal that there are at least two species of positive permission, static and dynamic. Static positive permission guides the citizen in the deontic assessment of specific actions, and behaves like a weakened obligation. Dynamic positive permission guides the legislator by describing the limits on what may be prohibited without violating static permissions. It behaves like a strengthened negative permission in may respects, but differs in those in which the set G of explicit obligations is allowed to vary.

The investigation also brings out the importance, when studying normative codes, of representing openly both the explicit obligations and the explicit permissions. If this is not done, it becomes difficult to separate negative from positive permission in any more than intuitive terms; impossible to articulate the difference between the two different kinds of positive permission, static and dynamic; and equally impossible to describe in formal terms the difference between dynamic and negative permission.

The conceptual relationships between these kinds of permission are shown in Figure 1, as a tree of types and subtypes of permission. The for-

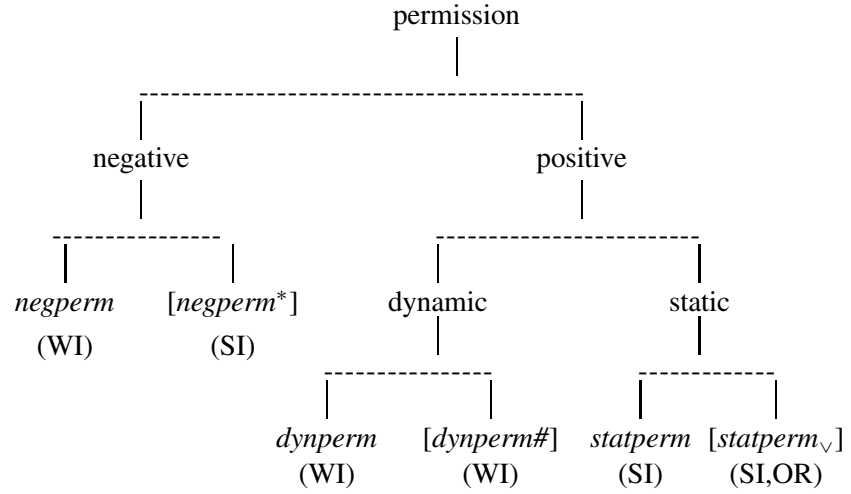

Figure 1.

TABLE I

$out(G) \subseteq statperm(P, G)$
$out(G) \subseteq^{c} dynperm(P, G)$
$out(G) \subseteq^{c} negperm(P, G)$ iff G is internally coherent
$statperm(P, G) \subseteq^{c} dynperm(P, G)$
$statperm(P, G) \subseteq^{c} negperm(P, G)$ iff G is cross-coherent with P
$dynperm(P, G) \subseteq negperm(P, G)$ iff G is cross-coherent with P

mally defined operations are in italics at the leaves; intuitive concepts are in roman at the non-leaf nodes. Below each formal operation we recall its satisfaction of the rules SI, WI, OR. The three operations between braces were defined *en passant* without further study – $statperm_{\vee}$ at the end of Section 4, $negperm^{*}$ in note 5, *dynperm#* in note 7.

In Table I we recall the inclusion relations (full, partial, conditional) between the output operation and the three permission operations studied, which were noted section by section in the paper. The three important inclusions are those of lines 1, 4, 6. The others follow from them immediately, but we list them for memory.

ACKNOWLEDGEMENTS

An earlier version of this paper was presented at the conference DEON'02 held in Imperial College London 22–24 May 2002. The authors would like to thank a referee of the DEON'02 conference for vigorous comments, and a referee of the *JPL* for further suggestions. Jan Broersen, Andrew Jones and Henry Prakken also made valuable remarks.

The work for the paper was carried out while the second author was at the Department of Artificial Intelligence, Vrije Universiteit Amsterdam, The Netherlands.

APPENDIX 1: INPUT/OUTPUT OPERATIONS

We briefly recall the central concepts of the theory of input/output operations. We work in a Boolean context, that is, a propositional language closed under the usual truth-functional connectives. The central objects of attention are ordered pairs (a, x) of formulae, which we read forwards. Intuitively, we think of each pair (a, x) as a rule, with body a representing

a possible *input*, and head x for a corresponding *output*. We call a set G of such pairs a *generating set*. The letter G also serves as a reminder of the interpretation (among others) of the pairs as conditional goals or obligations. When A is a set of formulae, we write $G(A)$ for its image under G, i.e. $G(A) = \{x : (a, x) \in G \text{ for some } a \in A\}$.

The operation *out*(G,A) takes as argument a generating set G, and an input set A of formulae, delivering as value an output set of formulae. We focus on four operations, which we define explicitly by equations. In so far as the definitions make no appeal to derivations or inductive processes, they may be thought of as semantic in a broad sense of the term.

- *Simple-minded output*: $out_1(G, A) = Cn(G(Cn(A)))$.
- *Basic output*: $out_2(G, A) = \bigcap\{Cn(G(V)) : A \subseteq V, V \text{ complete}\}$.
- *Reusable simple-minded output*: $out_3(G, A) = \bigcap\{Cn(G(B)) : A \subseteq B = Cn(B) \supseteq G(B)\}$.
- *Reusable basic output*: $out_4(G, A) = \bigcap\{Cn(G(V)) : A \subseteq V \supseteq G(V), V \text{ complete}\}$.

Here, *Cn* is classical consequence. A *complete* set is one that is either maxiconsistent or equal to the set of all formulae of the language. When A is a singleton $\{a\}$, we write $out_i(G, a)$ for $out_i(G, \{a\})$.

We have the inclusions $out_1(G, A) \subseteq \{out_2(G, A), out_3(G, A)\} \subseteq out_4(G, A) \subseteq Cn(A \cup m(G))$, but not in general conversely. Here $m(G)$ is the set of all materialisations of elements of G, i.e. the set of all formulae $b \rightarrow y$ with $(b, y) \in G$. In none of these four systems are inputs automatically outputs, that is, we do not in general have $A \subseteq out(G, A)$. Nor do the systems validate contraposition: we may have $x \in out(G, a)$ without $\neg a \in out(G, \neg x)$.

For each of these four principal operations, we may also consider a throughput version that also allows inputs to reappear as outputs. These are the operations $out_i^+(G, A) = out_i(G^+, A)$, where $G^+ = G \cup I$ and I is the set of all pairs (a, a) for formulae a.

It turns out that $out_4^+ = Cn(A \cup m(G))$, thus collapsing into classical logic. $Out_3^+(G, A)$ does not collapse in this way, but may be expressed more simply as $\bigcap\{B : A \subseteq B = Cn(B) \supseteq G(B)\}$.

These operations are distinct, with the exception that $out_2^+ = out_4^+$. This identity may be verified as follows. The left-in-right inclusion is immediate. To show the converse, suppose $x \notin out_2^+(G, A)$. Then there is a complete set V with $A \subseteq V$ and $x \notin Cn(G^+(V))$. To prove that $x \notin out_4^+(G, A)$ we need only show that $G^+(V) \subseteq V$. But $V \subseteq G^+(V)$ so if the converse fails then $G^+(V)$ is classically inconsistent so that $Cn(G^+(V))$ contains all propositions of the language, contradicting $x \notin Cn(G^+(V))$.

We write *out* without a subscript to cover indifferently all these seven distinct input/output operations. We move freely between the notations $x \in out(G, A)$ and $(A, x) \in out(G)$. The former is more useful when working directly with the above explicit definitions of the various kinds of output; the latter is more convenient when considering their characterizations using derivations, to which we now turn.

In derivations, we work with singleton inputs, defining derivability from an input set A as derivability from the conjunction of finitely many elements of A. For any set of derivation rules, we say that a pair (a, x) of formulae is *derivable from G using those rules*, and write $(a, x) \in deriv(G)$, iff (a, x) is in the least set that includes G and is closed under the rules. The specific rules considered are:

TAUT (tautologies):	From no premises to (t, t) for any tautology t
SI (strengthening the input):	From (a, x) to (b, x) whenever $a \in Cn(b)$
AND (conjunction of output):	From (a, x), (a, y) to $(a, x \wedge y)$

WO (weakening output):	From (a, x) to (a, y) whenever $y \in Cn(x)$
OR (disjunction of input):	From (a, x), (b, x) to $(a \vee b, x)$
CT (cumulative transitivity):	From (a, x), $(a \wedge x, y)$ to (a, y).

Here again, we emphasize, *Cn* is classical consequence. The rule TAUT is merely a technical one, to cover a limiting case. For when t is a tautology, the semantic definition gives us $t \in out(G, a)$ even when G is empty. To derive (a, t) from G it suffices to apply TAUT and SI.

As shown in (Makinson and van der Torre, 2000), simple-minded output coincides with derivability using TAUT, SI, AND, WO; basic output to those plus OR; simple-minded reusable output to the first four plus CT; and reusable basic output to all six. In other words, $x \in out(G, a)$ iff $(a, x) \in deriv(G)$, where the rules defining *deriv* are those mentioned as corresponding to *out*. For the augmented throughput versions, authorising inputs to reappear as outputs, add the zero-premise rule:

ID: From no premises to (a, a).

All of our systems of derivation admit the rules SI and WO, and so satisfy replacement of input, and of output, by classically equivalent propositions. That is, if $(a, x) \in deriv(G)$ then $(a', x') \in deriv(G)$ whenever $Cn(a) = Cn(a')$ and $Cn(x) = Cn(x')$. In derivations, it is convenient to treat replacement of logically equivalent propositions as a 'silent rule' that may be applied at any step without explicit justification.

APPENDIX 2: PROOF OF OBSERVATION 3

We prove that the non-repetition property holds for out_1, out_2, out_3 (under the rules specified), and conjecture a counterexample for out_4.

We need several lemmas. In each of them it is important to distinguish between a node $n : (a, x)$ of a derivation and the pair (a, x) that it carries, since distinct nodes may carry the same pair. We also recall from Appendix 1 that in derivation systems for input/output operations, replacement of classically equivalent propositions is treated as a 'silent rule' that may be used at any stage in a derivation without explicit mention. For example, in the proof below of part (c) of Observation 3, when $w \in Cn(z)$ we may replace $z \wedge w$ by z in the head of a pair $(c, z \wedge w)$, getting (c, z). We use the sign $\approx$ for classical equivalence.

The first lemma is needed for the cases out_1, out_2. It tells us that in a derivation using at most TAUT, SI, WO, AND, we can eliminate all the leaves carrying a given pair (a, x) provided we disjoin $\neg x$ with the head of each of its nodes.

LEMMA 3.1.1. *Let D be any derivation using at most* TAUT, SI, WO, AND *with root r: (b, y) and leaf-set L. Let n: $(a, x) \in L$. Then there is a derivation D' with root r': $(b, \neg x \vee y)$, using the same rules, from a subset of the same leaves plus possibly a leaf carrying (t, t), such that (a, x) does not decorate any leaf of D'.*

Proof sketch. Straightforward induction on the derivation. □

The second lemma is also needed for the cases out_1, out_2. It tells us that a derivation using at most TAUT, SI, WO, AND never diminishes the power of a body.

LEMMA 3.1.2. *Let D be any derivation using at most* TAUT, SI, WO, AND. *Then the body of the root classically implies the body of each leaf.*

Proof sketch. Straightforward induction on the derivation. □

The next lemma is needed for the case out_2. It is a 'phasing lemma' for OR.

LEMMA 3.2. *Let D be any derivation using at most* TAUT, SI, WO, AND, OR. *Then there is a derivation D' of the same root from a subset of the same leaves, that applies* OR *only at the end* (*i.e. no application of* OR *is followed by any application of the other rules*).

Proof sketch. Straightforward induction on the derivation. Also follows immediately from the more powerful phasing Theorem 19(b) of (Makinson and van der Torre, 2000). □

Our last three lemmas are needed for the case of out_3. The first is another phasing lemma, for the rule CTA. This is the rule authorizing passage from (a, x) and $(a \wedge x, y)$ to $(a, x \wedge y)$. Conceptually, it is just CT and AND put together; proof-theoretically it behaves quite differently.

LEMMA 3.3.1. *Let D be any derivation using at most* TAUT, SI, WO, CTA. *Then there is a derivation D' of the same root from a subset of the same leaves, in which rules are applied in the order* TAUT, SI, CTA, WO.

Proof sketch. By induction on the derivation. In the induction step, note that the rule CTA is under consideration, rather than the two rules CT and AND. We remark that this lemma is closely related to Theorem 19(c) of (Makinson and van der Torre, 2000). □

The next lemma puts the rule CTA under the microscope. We distinguish between the asymmetric premises by calling (a, x) the *minor premise* and $(a \wedge x, y)$ the *major premise*.

LEMMA 3.3.2. *Any succession of applications of* CTA *may be replaced by a succession in which no major premise of an application of* CTA *is the conclusion of another application of* CTA.

Proof sketch. Suppose we have two applications of CTA violating the desired pattern:

$$\begin{array}{ccc} (a, x) & (a \wedge x, y) & (a \wedge x \wedge y, z) \\ | & \text{----------} & \text{CTA} \\ | & (a \wedge x, y \wedge z) & \\ \text{----------} & & \text{CTA} \\ (a, x \wedge y \wedge z) & & \end{array}$$

We may replace this by the permuted applications:

$$\begin{array}{ccc} (a, x) & (a \wedge x, y) & (a \wedge x \wedge y, z) \\ \text{----------} & \text{CTA} & | \\ (a, x \wedge y) & & | \\ \text{----------} & & \text{CTA} \\ (a, x \wedge y \wedge z) & & \end{array}$$

Finally, we need a lemma telling us that a derivation using at most TAUT, SI, AND, CTA never diminishes the power of a head. The same is also true for OR, so we include it too even though we do not need it for present purposes.

LEMMA 3.3.3. *Let D be any derivation using at most* TAUT, SI, AND, CTA, OR. *Then the head of the root classically implies the head of each leaf.*

Proof sketch. Straightforward induction on the derivation. □

OBSERVATION 3. *The non-repetition property holds for:*

(a) out_1 *with its usual rules* TAUT, SI, WO, AND,
(b) out_2 *with its usual rules, i.e. the above plus* OR,
(c) out_3 *with the rules* TAUT, SI, WO, CTA.

Proof of (a). Consider any derivation D using at most the rules TAUT, SI, WO, AND with root r: (b, y) and leaf-set L. If no leaf of D carries (a, x) we are done. Suppose that at least one leaf of D carries (a, x). By Lemma 3.1.1 there is a derivation D' with root r': $(b, \neg x \vee y)$, using the same rules from a subset of the same leaves plus possibly a leaf carrying (t, t), such that (a, x) does not decorate any leaf of D'. Now add (a, x) as a new leaf to D'. By Lemma 3.1.2, since (a, x) figured as a leaf of D, $a \in Cn(b)$. Hence in the new derivation we may apply SI to get (b, x), and then apply AND, WO to this with r' to get (b, y).

Proof of (b). The argument is essentially the same as for case (a), with suitable additions. In detail, consider any derivation D using at most the rules TAUT, SI, WO, AND, OR, with root r: (b, y). By Lemma 3.2 we can phase it with all applications of OR at the end. So we have a number of derivations D_i without OR, with roots r_i: (b_i, y), such that $b \approx \vee b_i$. The applications of OR may clearly be re-ordered so that we apply OR first to the pairs (b_j, y) such that (a, x) is carried by a leaf of D_j. By Lemma 3.1.1 each such D_j can be replaced by a derivation D'_j of $(b_j, \neg x \vee y)$, using the same rules, from a subset of the same leaves plus possibly a leaf carrying (t, t), such that (a, x) does not appear as a leaf of D_j. Apply OR to the roots of those derivations alone, getting r': $(\vee b_j, \neg x \vee y)$. Now add (a, x) as a leaf. By Lemma 3.1.2, since (a, x) figured as a leaf of D_j, we have $a \in Cn(b_j)$, so $a \in Cn(\vee b_j)$. Hence in the new derivation we may apply SI to the new leaf (a, x) to get $(\vee b_j, x)$, and then apply AND and WO to this with r' to get $(\vee b_j, y)$. Then apply OR to this with the roots of the remaining D_i in which (a, x) did not occur as a leaf, and we are done.

Proof of (c). Consider any derivation D using at most the rules TAUT, SI, WO, CTA with root r: (b, y). By Lemmas 3.3.1 and 3.3.2 we can phase it with rules applied in the order TAUT, SI, CTA, WO, and with the major premise of an application of CTA never serving as the conclusion of another application of CTA. By Observation 3(a), already established above, we may also assume that for each node whose subtree does not involve CTA, that subtree contains at most one leaf carrying the pair (a, x). Call this derivation D'.

Suppose that in D' the pair (a, x) decorates at least two distinct leaves. We show that we can eliminate one of them. Let n be a first node of the derivation whose subtree has that property (if there is more than one such first n, choose any one of them). Clearly, n must be the conclusion of an application of CTA. We thus have the step:

$$\begin{array}{l} p\colon (c, z) \qquad\qquad q\colon (c \wedge z, w) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{ CTA} \\ \qquad\quad n\colon (c, z \wedge w) \end{array}$$

where (1) the subtree determined by p uses at most TAUT, SI, CTA and contains a leaf carrying (a, x), and (2) the subtree determined by q uses at most TAUT, SI and also contains a leaf carrying (a, x). Applying Lemma 3.3.3 to (1) gives us $x \in Cn(z)$. From (2) it follows that the subtree determined by q has (a, x) as its sole leaf and uses only SI,

so that $x \approx w$. Since $x \in Cn(z)$ we thus have $z \approx z \wedge w$. Hence we may delete from the tree the node n and the subtree determined by q, and we are done. □

This completes the proof of Observation 3. It seems that the result cannot be extended to out_4 (under the rules TAUT, SI, WO, CTA, OR). Consider the derivation of $((a \wedge (\neg x \vee b)) \vee (b \wedge (\neg y \vee a)), x \wedge y)$ from (a, x), (b, y) below:

$$\begin{array}{cccc} (a,x) & (b,y) & (b,y) & (a,x) \\ \mid \text{SI} & \mid \text{SI} & \mid \text{SI} & \mid \text{SI} \\ (a \wedge (\neg x \vee b), x) & (a \wedge x \wedge b, y) & (b \wedge (\neg y \vee a), y) & (b \wedge y \wedge a, x) \\ \multicolumn{2}{c}{\text{---------- CTA}} & \multicolumn{2}{c}{\text{---------- CTA}} \\ \multicolumn{2}{c}{(a \wedge (\neg x \vee b), x \wedge y)} & \multicolumn{2}{c}{(b \wedge (\neg y \vee a), x \wedge y)} \\ \multicolumn{4}{c}{\text{---------- OR}} \\ \multicolumn{4}{c}{(a \wedge (\neg x \vee b)) \vee (b \wedge (\neg y \vee a)), x \wedge y)} \end{array}$$

In this derivation, each premise is used twice. It is easy to show that there is no derivation in which no premise is used more than once. We conjecture, more strongly, that there is no derivation in which premise (a, x), say, is used only once, no matter how many times the other premise is used.

NOTES

[1] In the logical literature, the distinction between positive and negative permission seems first to have been made by (von Wright, 1959) and in more detail in (von Wright, 1963). For a critical discussion, see (Alchourrón and Bulygin, 1984).

[2] As we do not consider the imposition of consistency constraints on the operations, no familiarity is needed with (Makinson and van der Torre, 2001).

[3] Thus when x is negatively permitted with respect to a, this does not mean that a is a sufficient condition for x to be permitted. It means that *a is not a sufficient condition for x to be prohibited*. This point was made forcefully by (Alchourrón, 1993, Section 3.4.1.1). He attributes it to Hector-Neri Castañeda, but without giving a specific reference, and the authors have not been able to locate a source in Castañeda's publications. As Alchourrón puts the point, the negation of a conditional is not in general a conditional, and in particular the negation of a conditional obligation is not a conditional permission, whether positive or negative. For this reason, he regards the very term 'negative conditional permission' as misleading, and recommends its abandonment. We sympathize with this recommendation, but have not followed it, partly because the term is so well established and partly because it is difficult to devise a less misleading one to replace it.

[4] One may imagine strengthened forms of negative permission that satisfy SI, so that the body may be regarded as a sufficient condition for the head. For example, as suggested to the authors by Jan Broersen, one may put $(a, x) \in negperm^*(G)$ iff there is no consistent b with $a \in Cn(b)$ such that $(b, \neg x) \in out(G)$, i.e. iff $(b, x) \in negperm(G)$ for every consistent b with $a \in Cn(b)$. Evidently, this operation satisfies SI for consistent bodies. We do not study it in this paper. It should not be confused with the operation of *dynperm*, which we define in Section 3 and relate to *negperm* in Section 6. In particular, *dynperm* satisfies WI rather than SI.

[5] The empty set would not suffice, since all the rules in $(HR)^{-1}$ have a permission-premise. In the case that G is internally coherent in the sense defined early in Section 2, then as already noted we have $out(G) \subseteq^c negperm(G)$, so that $out(G)$ might be suggested as a suitable basis. However, it is easy to see that in general this does not suffice. Take the case that $G = \varnothing$ and out is any out_i for $i \in \{1, 2, 3, 4\}$. Then $out(G)$ is the set of all pairs (a, t), where t is a tautology, while $negperm(G)$ is the larger set of all pairs (a, x) with $x \neq f$, where f is any contradiction. Application of the rules TRIV, WI, WO, AND does not lead us out of the former set, and so does not take us to the latter one.

[6] We do not consider here the contractions or revisions that one might wish to make to the code when A is inconsistent with some $z \in Z$. This is a separate matter, and forms part of the logic of normative change.

[7] This way of expressing the definition suggests a further concept of *proper dynamic* permission. We may wish to require that the addition of $(a, \neg x)$ to G not only *leaves us with* a cross-incoherent system, but also *creates* the cross-incoherence. One way of attempting to express this formally would be to say: $(a, x) \in dynperm\#(P, G)$ iff (as before) $G \cup \{(a, \neg x)\}$ is not cross-coherent with P and (in addition) G is cross-coherent with P. This notion is certainly of interest, but given the space limitations of this paper, we do not study its properties here. Our methodology is: study the simple components first and their compounds afterwards.

[8] We have chosen the names *static* and *dynamic* permission because in the case of dynamic permission we are considering what would happen if we were to change the current code by adding a certain prohibition, whereas in the case of static permission we consider the current code alone. From a purely formal point of view, one could speak of *forward* versus *backward* permission; from a social point of view one could contrast *citizens*' with *legislators*' permission.

[9] For a counterexample in one direction, put $G = \varnothing$ and $P = \{(a, x)\}$, where a, x, y are distinct elementary letters. Then $(a, y) \in negperm(G)$ since $(a, \neg y) \notin out(G)$, but $(a, y) \notin statperm(P, G) = out(\{(a, x)\})$. For a counterexample in the other direction, put $G = \{(a, x)\}$ and $P = \{(a, \neg x)\}$. Then $(a, \neg x) \in statperm(P, G) = out(\{(a, x), (a, \neg x)\})$, but $(a, \neg x) \notin negperm(G)$ since $(a, x) \in out(G)$.

[10] In other words, the inclusion $statperm(P, statperm(P, G)) \subseteq statperm(P, G)$ can fail. Witness the example $G = \varnothing$, $P = \{(a, x), (a, \neg x)\}$, so that $(a, f) \notin statperm(P, G) = out(\{(a, x)\}) \cup out(\{(a, \neg x)\})$ while $(a, f) \in statperm(P, statperm(P, G)) = out(\{(a, x), (a, \neg x)\})$.

[11] For a counterexample to idempotence of *dynperm*, put $G = \{(a, x)\}$ and $P = \{(b, y)\}$. Then $(a, x \wedge y)$ is in $dynperm(dynperm(P, G), G)$ but not in $dynperm(P, G)$. To check the former, note that since $(b, y) \in P$, we have $(a \wedge b, y) \in statperm(P, G)$, so that $(a, y) \in dynperm(P, G) \subseteq statperm(dynperm(P, G), G)$, so finally using the presence of (a, x) in G, $(a, x \wedge y) \in dynperm(dynperm(P, G), G)$. But although $(a, y) \in dynperm(P, G)$, we have $(a, y) \notin statperm(P, G)$ and $(a, x \wedge y) \notin dynperm(P, G)$. We note that idempotence would hold if the definition of *dynperm* were narrowed to replace $statperm(P, G)$ in it by P; but this would not accord well with the intuitive motivation given in Section 3.

REFERENCES

Alchourrón, C. E. (1993): Philosophical foundations of deontic logic and the logic of defeasible conditionals, in J. J.Meyer and R. J. Wieringa (eds), *Deontic Logic in Computer Science: Normative System Specification*, Wiley, New York.

Alchourrón, C. E. and Bulygin, E. (1984): Permission and permissive norms, in W. Krawietz et al. (eds), *Theorie der Normen*, Duncker & Humblot, Berlin.

Makinson, D. (1999): On a fundamental problem of deontic logic, in P. McNamara and H. Prakken (eds), *Norms, Logics and Information Systems. New Studies in Deontic Logic and Computer Science*, Frontiers Artif. Intell. Appl. 49, IOS Press, Amsterdam, pp. 29–53.

Makinson, D. and van der Torre, L. (2000): Input/output logics, *J. Philos. Logic* **29**, 383–408.

Makinson, D. and van der Torre, L. (2001): Constraints for input/output logics, *J. Philos. Logic* **30**, 155–185.

Makinson, D. and van der Torre, L. (2003): What is input/output logic?, in *Foundations of the Formal Sciences II: Applications of Mathematical Logic in Philosophy and Linguistics*, Trends in Logic Series 17, Kluwer Academic Publishers, Dordrecht, pp. 163–174.

Von Wright, G. (1959): On the logic of negation, *Soc. Scient. Fennica Com. Physico-Math.* XXII, 4.

Von Wright, G. (1963): *Norm and Action*, Routledge, London.

DAVID MAKINSON
Department of Computer Science
King's College London
Strand Campus, London WC 2R 2LS,
United Kingdom
E-mail: makinson@dcs.kcl.ac.uk

LEENDERT VAN DER TORRE
CWI, Kruislaan 413
PO Box 94079
1090 GB Amsterdam,
The Netherlands
E-mail: torre@cwi.nl