Journal of Philosophical Logic 29 (2000) 383-408.

DAVID MAKINSON AND LEENDERT VAN DER TORRE

INPUT/OUTPUT LOGICS

ABSTRACT. In a range of contexts, one comes across processes resembling inference, but where input propositions are not in general included among outputs, and the operation is not in any way reversible. Examples arise in contexts of conditional obligations, goals, ideals, preferences, actions, and beliefs. Our purpose is to develop a theory of such input/output operations. Four are singled out: simple-minded, basic (making intelligent use of disjunctive inputs), simple-minded reusable (in which outputs may be recycled as inputs), and basic reusable. They are defined semantically and characterised by derivation rules, as well as in terms of relabeling procedures and modal operators. Their behaviour is studied on both semantic and syntactic levels.

KEY WORDS: input/output logic, reusability, identity, conditional goals, conditional obligations, deontic logic

1. INTRODUCTION

Imagine a black box into which we may feed propositions as input, and that also produces propositions as output. Of course, classical consequence may itself be seen in this way, but it is a very special case, with additional features - inputs are also themselves outputs, since any proposition classically implies itself, and the operation is in a certain sense reversible, since contraposition is valid. However, there are many examples without those features. Roughly speaking, they are of two main kinds.

The box may stop some inputs, while letting others through, perhaps in modified form. Inputs may record reports of agents, of the kind 'according to source *i*, *x* is true', while the box may give as output either *x* itself, a qualified version of *x*, or nothing at all, according to the identity of *i*. Or it might give output *x* only when at least two distinct sources vouch for it, and so on. Inputs might be facts about the performance of the stock-market today, and outputs an analyst's commentary; or facts about your date and place of birth, with output your horoscope readings. In these examples, the outputs express some kind of belief or expectation.

Again, inputs may be conditions, with outputs expressing what is deemed desirable in those conditions. The desiderata may be obligations of a normative system, ideals, goals, intentions or preferences. In general, a fact entertained as a condition may itself be far from desirable, so that inputs are not always outputs; and as is widely recognised, contraposition is inappropriate for conditional goals.

Our purpose is to develop a general theory of propositional input/output operations, covering both kinds of example. Particular attention is given to the case where outputs

may be recycled as inputs. In a companion paper (Makinson and van der Torre, to appear), we examine the imposition of constraints on output.

From a very general perspective, logic is often seen as an 'inference motor', with premises as inputs and conclusions as outputs (figure 1). But it may also be seen in another role, as 'secretarial assistant' to some other, perhaps non-logical, transformation engine (figure 2). From this point of view, the task of logic is one of preparing inputs before they go into the machine, unpacking outputs as they emerge and, less obviously, co-ordinating the two. The process as a whole is one of 'logically assisted transformation', and is an inference only when the central transformation is so. This is the general perspective underlying the present paper. It is one of 'logic at work' rather than 'logic in isolation'; we are not studying some kind of non-classical logic, but a way of using the classical one.

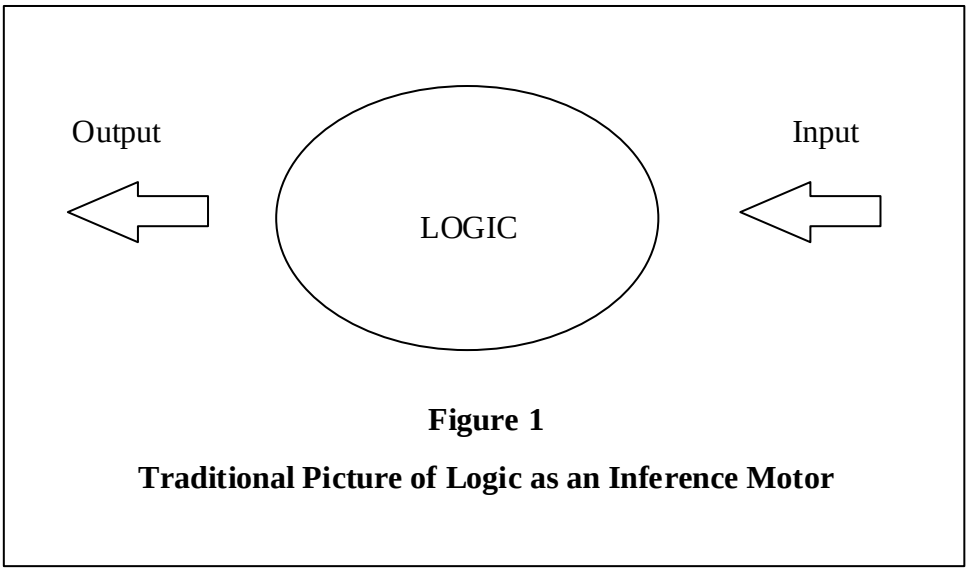


Figure 1

Traditional Picture of Logic as an Inference Motor

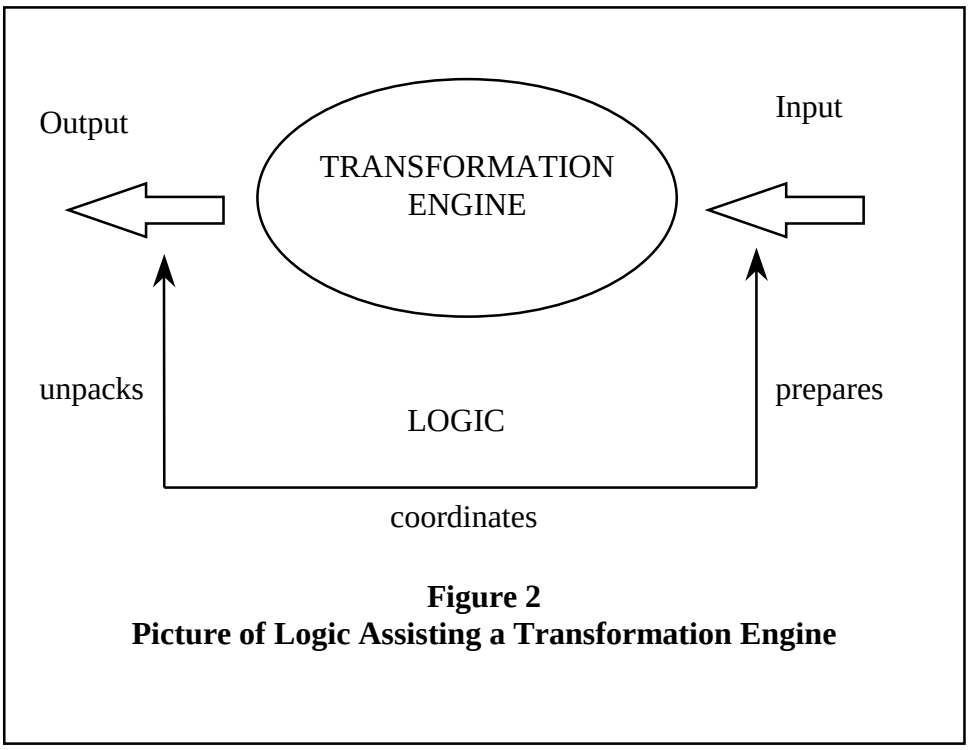


Figure 2
Picture of Logic Assisting a Transformation Engine

On a pre-logical level, this picture is perfectly familiar from elementary set theory. Consider any universe L, not necessarily of propositions, and any relation $G \subseteq L^2$. For example, L may be the set of humans, and G the parent/child relation. Given an input $A \subseteq L$, the output of A under G may be understood simply as $G(A) = \{x: (a,x) \in G$ for some $a \in A\}$ - in the example, the set of all children of persons in A.

The present paper may be seen as investigating what happens to this basic picture when we pass to the logical level, i.e. when L is the set of propositions of some language, and input and output are both under the sway of the operation Cn of classical consequence. These are in a certain sense frills, but give rise to subtle and interesting behaviour.

2. LOGICAL LEVEL: THE PROBLEM

Consider a propositional language L, closed under at least the usual truth-functional connectives; its elements are called *formulae*. Let G be a set of ordered pairs (a,x) of formulae in L; the letter chosen serves as reminder of the interpretation (among others) of the pairs as conditional goals. We call G a *generating set*. We read a pair (a,x) forwards, i.e. with a as body and x as head; and we call the corresponding truth-functional formula $a \rightarrow x$ its *materialisation*, echoing the old name 'material implication' for the connective involved.

Suppose that we are also given a set A of formulae. Our problem is: how may we reasonably define the set of propositions x making up the output of A under G, or one might also say, of G given A, which we write $out(G,A)$? Alternatively, suppose we are given only the generating set G: how may we define the set of input/output pairs (A,x) arising from G, which we write $out(G)$?

These questions are the same, for we may define $(A,x) \in out(G)$ iff $x \in out(G,A)$ or conversely. But the two formulations give a rather different *gestalt*, and one is sometimes more convenient rather than the other. As we shall see, the latter tends to be clearer in semantic contexts, whilst the former is easier to work with when considering derivations in a syntactic context. We shall move freely from one to the other, just as one moves between Cn and $\vdash$ for classical consequence.

3. SIMPLE-MINDED OUTPUT

3.1. Semantic definition

The simplest response to our problem is to put $out(G,A) = Cn(G(Cn(A)))$, where the function $G(.)$ is defined as on the pre-logical level, and Cn alias $\vdash$ is classical consequence. In other words, given a set A of formulae as input, we first collect all of its consequences, then apply G to them, and finally consider all of the consequences of what is thus obtained (figure 3).

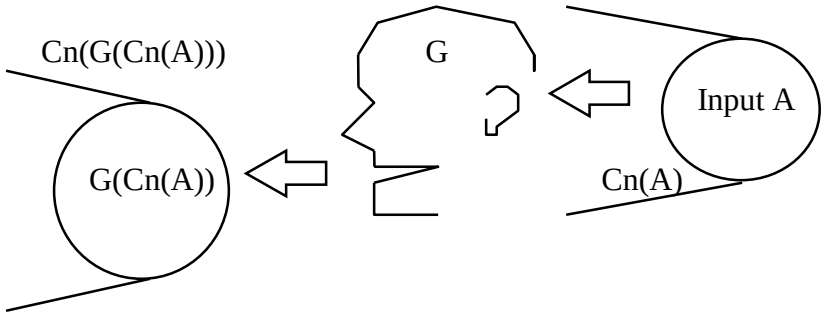


Figure 3
Simple-Minded Output: $out_1(G,A) = Cn(G(Cn(A)))$

Under this definition, which we call *simple-minded output* and write as $out_1(G,A)$, inputs are not in general outputs; that is, we do not have $A \subseteq out_1(G,A)$.

Example 1. Put $G = \{(a,x)\}$where a,x are distinct elementary letters, and put $A = \{a\}$. Then $G(Cn(a)) = \{x\}$ so $a \notin out_1(G,a) = Cn(G(Cn(a))) = Cn(x)$. Contraposition also fails, for although $x \in out_1(G,a)$ we have $\neg a \notin out_1(G,\neg x)$: since $a \notin Cn(\neg x)$ we have $G(Cn(\neg x)) = \varnothing$ so that $\neg a \notin out_1(G,\neg x) = Cn(G(Cn(\neg x))) = Cn(\varnothing)$.

Clearly, this operation is inadequate for some purposes, for it is unable to handle disjunctive inputs intelligently.

Example 2. Put $G = \{(a,x), (b,x)\}$and $A = \{a \vee b\}$. Then $Cn(A) \cap b(G) = \varnothing$ where we write $b(G)$ for the set of all bodies of elements of G, i.e. in this example the set $\{a,b\}$. Hence also $G(Cn(A)) = \varnothing$ so that $Cn(G(Cn(A))) = Cn(\varnothing)$. However, in many contexts we would want to put x in the output, as it can be obtained from each of the two disjuncts of the input.

Nevertheless, the operation of simple-minded output has an interest of its own, and its study also helps prepare the way for more sophisticated ones.

3.2. Syntactic characterisation

Our definition of simple-minded output is, in a broad sense of the term, semantic. It is not difficult to give it a syntactic characterisation in terms of derivation rules.

In general, for any set of rules, we say that a pair (a,x) of formulae is *derivable using those rules* from a set G of such pairs iff (a,x) is in the least set that includes G, contains the pair (t,t) where t is a tautology, and is closed under the rules. In the systems studied here, it will make no difference which tautology t is chosen. Our

notations are $(a,x) \in deriv(G)$ or equivalently $x \in deriv(G,a)$, with a subscript to indicate the set of rules employed.

When A is a set of formulae, derivability of (A,x) from G is defined as derivability of (a,x) from G for some conjunction $a = a_1 \wedge \ldots \wedge a_n$ of elements of A. We understand the conjunction of zero formulae to be a tautology, so that $(\varnothing,x)$ is derivable from G iff (t,x) is for some tautology t.

In the particular case of simple-minded output, we use the following three rules determining an operation $deriv_1$. Of these, the first governs the use of inputs (strengthening the input: SI), while the other two deal with the management of outputs (conjunction in the output: AND; weakening the output: WO).

SI: From (a,x) to (b,x) whenever $b \vdash a$
AND: From (a,x), (a,y) to $(a,x \wedge y)$
WO: From (a,x) to (a,y) whenever $x \vdash y$.

OBSERVATION 1. $Out_1(G,A) = deriv_1(G,A)$.

Outline of proof. The inclusion from right to left is straightforward by induction on length of derivation. From left to right, suppose $x \in Cn(G(Cn(A)))$. Then by compactness of Cn there are $x_1,\ldots,x_n \in G(Cn(A))$ with $x \in Cn(x_1 \wedge \ldots \wedge x_n)$. In the case that $n = 0$, x is a tautology t and we can also put $a = t$ giving us a one-step derivation of (t,t). In the case that $n \neq 0$ we proceed as follows. For each $i \leq n$, since $x_i \in G(Cn(A))$ there is a $b_i \in Cn(A)$ with $(b_i,x_i) \in G$. Putting $b = b_1 \wedge \ldots \wedge b_n$ we note that $b \in Cn(A)$, and so by compactness $b \in Cn(a)$ for some conjunction $a = a_1 \wedge \ldots \wedge a_m$ of elements of A. We can thus construct a derivation whose leaves are the pairs (b_i,x_i), followed by applications of SI to get the pairs (a,x_i), followed by applications of AND to get $(a, x_1 \wedge \ldots \wedge x_n)$, followed finally by WO to get (a,x).

Evidently, the proof of Observation 1 also provides a 'universal order' for derivations of simple-minded output: SI, AND, WO. More on this in section 8.

4. BASIC OUTPUT

4.1. Semantic definition and syntactic characterisation

As already remarked, simple-minded output is unable to process disjunctive inputs intelligently. How may this be done? On the syntactic level, the answer is obvious: define $deriv_2(G)$ by adding the following rule to those for simple-minded derivations:

OR: From (a,x), (b,x) to $(a \vee b,x)$.

On the semantic level, we define *basic output*, $out_2(G,A)$, as $\cap\{Cn(G(V)): v(A) = 1\}$, in the principal case that A is classically consistent (see figure 4). Here, v ranges over boolean valuations and $V = \{b: v(b) = 1\}$. In the limiting case that there is no such v (which by classical logic happens iff A is inconsistent) we put $out_2(G,A)$ to be

$Cn(G(L))$ where L is the set of all boolean formulae; this equals $Cn(h(G))$ where $h(G)$ is the set of all heads of elements of G.

Equivalently: $out_2(G,A) = \cap\{Cn(G(V)): A \subseteq V, V \text{ complete}\}$. Here, by a *complete* set we mean one that is either maxiconsistent or equal to L. There is always at least one complete V that includes A, namely L, and so there is no need for a separate limiting case. The same trick could be done with the first formulation, by allowing v to be either a boolean valuation or the function that puts $v(b) = 1$ for all formulae b.

Note that as classical consequence Cn is monotonic, and the transformation $G(X)$ is also monotonic in each of X and G, both simple-minded and basic output are monotonic in each of their arguments.

To compare basic with simple-minded output, notice that simple-minded output can also be expressed as an intersection. Trivially, $out_1(G,A) = \cap\{Cn(G(B)): A \subseteq B = Cn(B)\}$. As is well known, $Cn(V) = V$ for any complete V, so we can say that basic output is like simple-minded output except that is restricts the choice of B to complete sets.

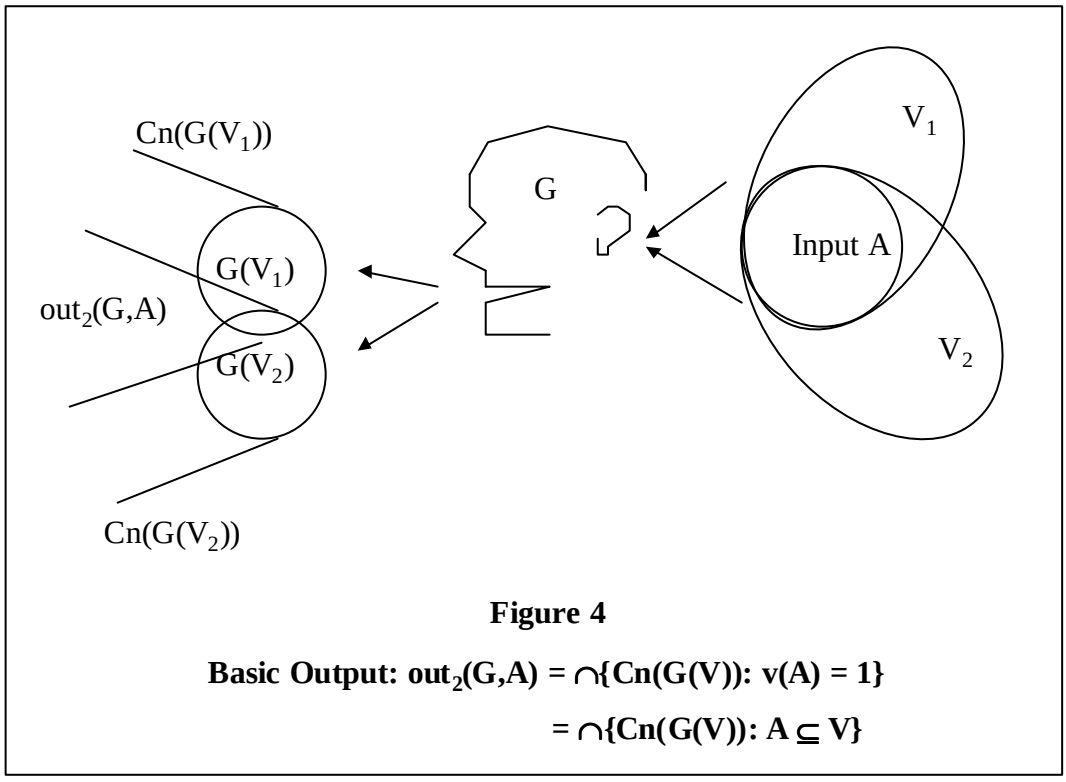


Figure 4

Basic Output: $out_2(G,A) = \cap\{Cn(G(V)): v(A) = 1\}$

$= \cap\{Cn(G(V)): A \subseteq V\}$

OBSERVATION 2. $Out_2(G,A) = deriv_2(G,A)$.

Outline of Proof. We begin by disposing of the limiting case that A is inconsistent. In that case $out_2(G,A) = Cn(h(G)) = deriv_2(G,A)$ by definition on the left and easy verification on the right. Next, we dispose of another limiting case, that $x \notin Cn(G(L))$.

Since L is complete and includes A, this gives immediately $x \notin out_2(G,A)$; and it is also easy to show by induction that $deriv_2(G,A) \subseteq Cn(G(L))$ so that $x \notin deriv_2(G,A)$. So consider finally the principal case that A is consistent and $x \in Cn(G(L))$.

The verification from right to left (soundness) is effected by first observing that it suffices to prove the result for individual formulae a and then carrying out a straightforward induction on length of derivation. The interesting case in the induction is that for the rule OR. Suppose $x \notin out_2(G,b \vee c)$. Then there is a boolean valuation v with $v(b \vee c) = 1$ and $x \notin Cn(G(V))$. But then either $v(b) = 1$ or $v(c) = 1$ so either $x \notin out_2(G,b)$ or $x \notin out_2(G,c)$.

For the converse (completeness), we can use a maximality argument, similar to that familiar for proving completeness in classical propositional logic, but with more verifications at each step. In sketch: suppose $x \notin deriv_2(G,A)$. Then by the monotony and compactness of the derivability operation in its right argument (both immediate from its definition) there is a maximal $A' \supseteq A$ with $x \notin deriv_2(G,A')$. It is easy to verify that A' is well-behaved with respect to conjunction and disjunction. Using the supposition $x \in Cn(G(L))$ we can also verify that it is well-behaved with respect to negation. Hence there is a boolean valuation v with $A' = V$. To complete the proof, one need only show that $x \notin out_1(G,V) = Cn(G(Cn(V))) = Cn(G(V))$ since V is closed under consequence. But this is immediate since by Observation 1 $out_1(G,V) = deriv_1(G,V) \subseteq deriv_2(G,V)$ and we have $x \notin deriv_2(G,V)$.

Evidently, Observation 2 implies the compactness of out_2, a fact rather difficult to verify directly from the semantic definition (in contrast to the situation for simple-minded output, where compactness is almost immediate).

We present two further characterisations of basic output. One uses relabeling of elementary letters, the other translates into modal logic. They have very similar structures. We regard these two characterisations as interesting curiosities more than useful tools, and they are not re-employed in subsequent sections. Hence sections 4.2 and 4.3 may be skipped without loss of continuity.

4.2. Account in terms of relabeling

The basic idea of this approach is to relabel the letters in the heads. This has the effect of isolating the heads from the bodies, so that information about one cannot be carried forwards or backwards to the other. Technically, alongside the existing language, introduce a fresh set of elementary letters, with one new letter p^* for each old letter p. For arbitrary old formulae x, define x^* in the natural way, by substituting the letters p^* for p in x. Write G^* for $\{b \rightarrow y^* : (b,y) \in G\}$, i.e. as the set of all materialisations of pairs (b,y^*) obtained by starring heads only of elements of G.

OBSERVATION 3. $x \in out_2(G,A)$ iff $x \in Cn(G(L))$ and $G^* \cup A \vdash x^*$.

Proof. We dispose of the limiting cases that A is inconsistent and that $x \notin Cn(G(L))$ in the same manner as for Observation 2. So suppose for the principal case that A is consistent and $x \in Cn(G(L))$.

Suppose first that the left side fails. Since A is consistent, there is a valuation v on unstarred letters with $v(A) = 1$ and $x \notin Cn(G(V))$. From the latter, there is a valuation w (also on unstarred letters) with $w(G(V)) = 1$ and $w(x) = 0$. Define a valuation w^* on formulae generated by starred letters by putting $w^*(p^*) = w(p)$ for each starred letter p^*. Write $v+w^*$ for the valuation on starred and unstarred letters determined by the two together. We claim that $v+w^*(G^* \cup A) = 1$ and $v+w^*(x^*) = 0$. The latter is immediate from $w(x) = 0$ since all letters in x^* are starred. Similarly, we have $v+w^*(A) = 1$ from $v(A) = 1$ since all letters in A are unstarred. It remains to check that $v+w^*(G^*) = 1$. Let $(b,y) \in G$ and suppose $v+w^*(b) = 1$; we need to show that $v+w^*(y^*) = 1$. Since b is unstarred the supposition tells us that $v(b) = 1$ so $b \in V$, so $y \in G(V)$ so by hypothesis $w(y) = 1$ so $w^*(y^*) = 1$ and finally $v+w^*(y^*) = 1$.

To show the converse, we could use the identity $out_2(G,A) = deriv_2(G,A)$ established by Observation 2, and proceed by induction on length of derivation, but we give a direct argument, as follows.

Suppose that the right side fails. Since we are assuming that $x \in Cn(G(L))$, there is a valuation defined on both starred and unstarred letters that satisfies $G^* \cup A$ and fails x^*. Without loss of generality, we may write this valuation as $v+w^*$ where v,w are defined on unstarred letters and w^* is defined from w as before. Thus $v+w^*(G^* \cup A) = 1$ and $v+w^*(x^*) = 0$. We show that $v(A) = 1$ and $x \notin Cn(G(V))$. We have $v(A) = 1$ immediately from $v+w^*(A) = 1$ since A contains only unstarred letters. For $x \notin Cn(G(V))$, it suffices to show that $w(x) = 0$ while $w(G(V)) = 1$. The former is immediate from $v+w^*(x^*) = 0$. For the latter, suppose $y \in G(V)$; we need to show $w(y) = 1$. Since $y \in G(V)$ there is an unstarred formula b with $(b,y) \in G$ and $b \in V$ so that $1 = v(b) = v+w^*(b)$. Since $v+w^*(G^*) = 1$ we have $v+w^*(b \rightarrow y^*) = 1$ so that $1 = v+w^*(y^*) = w^*(y^*) = w(y)$ as desired.

4.3. Modal formulation

The modal characterisation has strong formal parallels with the relabeling one. Its essential idea is to prefix heads with boxes and apply a suitable modal logic. Indeed any modal logic from a broad interval will do the job.

Consider the modal propositional language formed by adding a unary box operator to the classical language, and consider the modal calculus $\mathbf{K_0}$, serving as a lower bound on the interval, defined axiomatically as follows. Take as axioms all classical tautologies in that language and all formulae of the form $(a \rightarrow x) \rightarrow (\Box a \rightarrow \Box x)$; and take as rules passage from a, $a \rightarrow x$ to x (detachment), and passage from t to $\Box t$ for every classical tautology t. Evidently, we could reformulate the last rule as axioms $\Box t$ for every classical tautology t.

$\mathbf{K_0}$ is a subsystem of the familiar modal logic **K**; the latter also allows passage from a to $\Box a$ for every thesis a. We recall the well-known fact that for first-degree formulae (i.e. formulae without iteration of the box) all systems from $\mathbf{K_0}$ to **K45** agree.

Write G for the set of all modal formulae $b \rightarrow y$ with $(b,y) \in G$, and $Z \vdash_{\mathbf{S}} z$ to mean that $(\wedge Y \rightarrow z) \in \mathbf{S}$ for some finite $Y \subseteq Z$.

OBSERVATION 4. $x \in out_2(G,A)$ iff $x \in Cn(G(L))$ and $G \cup A \vdash_{\mathbf{S}} x$, for any modal logic **S** with $\mathbf{K_0} \subseteq \mathbf{S} \subseteq \mathbf{K45}$.

Proof. Since all systems from $\mathbf{K_0}$ to **K45** agree on first-degree formulae, we need only prove the Observation for **K**. In the limiting case that A is classically inconsistent both sides are equivalent to $x \in Cn(G(L))$ and we are done. So suppose that A is consistent.

Suppose $x \in out_2(G,A)$. Then by Observation 2, $(A,x) \in deriv_2(G)$ so we need only show by induction that whenever $(a,x) \in deriv_2(G)$ then $G \cup \{a\} \vdash_{\mathbf{K}} x$, which is straightforward.

Conversely, suppose $x \notin out_2(G,A)$. Since A is assumed consistent, there is a valuation v of boolean formulae with $v(A) = 1$ and $x \notin Cn(G(V))$. Fix one such v, and define a relational model (M,R,φ) by putting M to be the set of all purely boolean valuations and for $u,w \in M$ put $(u,w) \in R$ iff for every $(b,y) \in G$, if $u(b) = 1$ then $w(y) = 1$. Put $\varphi(w,p) = w(p)$ for all elementary letters p and all $w \in M$.

To complete the proof, it suffices to check that $\varphi(v, G \cup A) = 1$ while $\varphi(v, x) = 0$. Since $v(A) = 1$ and A is purely boolean, $\varphi(v,A) = 1$. Suppose $b \rightarrow y \in G$ and $\varphi(v,b) = 1$; then $(b,y) \in G$ and b is purely boolean so $v(b) = 1$ and also whenever $(v,w) \in R$ then by the definition of R, $w(y) = 1$; thus $\varphi(v, b \rightarrow y) = 1$. This shows $\varphi(v, G) = 1$. To show $\varphi(v, x) = 0$ we need to find a w with $(v,w) \in R$ and $\varphi(w,x) = 0$. But by hypothesis, $x \notin Cn(G(V))$ so there is a w with $w(G(V)) = 1$ and $w(x) = \varphi(w,x) = 0$. It remains only to check that $(v,w) \in R$. But if $(b,y) \in G$ and $v(b) = 1$ then immediately $y \in G(V)$ so $w(y) = 1$ and by the definition of R we are done.

5. REUSABLE OUTPUT

5.1. Idea and definitions

In certain situations, it may be appropriate for outputs to be available for recycling as inputs. For example, the elements (a,x) of G may be conditional norms of a kind that say that any configuration in which a is true is one in which x is desirable. In some contexts, we may wish to entertain hypothetically the items already seen as desirable, in order to determine what is in turn so. How may such a principle of reusability be expressed formally?

On the syntactic level, the answer again suggests itself naturally: add the following rule of 'cumulative transitivity' to those already available for simple-minded output, or those for basic output:

CT: From (a,x), $(a \wedge x,y)$ to (a,y).

Given SI, this immediately implies transitivity (from (a,x), (x,y) to (a,y)) but not conversely.

On the semantic level, we define *simple-minded reusable output*, written $out_3(G,A)$, as follows:

$$out_3(G,A) = \cap\{Cn(G(B)): A \subseteq B = Cn(B) \supseteq G(B)\}.$$

There is always at least one set B with $A \subseteq B = Cn(B) \supseteq G(B)$, namely L, and the intersection of any non-empty family of such sets satisfies the same condition.

Recalling again that simple-minded output can be expressed as an intersection, with $out_1(G,A) = \cap\{Cn(G(B)): A \subseteq B = Cn(B)\}$, we can say that reusable simple-minded output is like plain simple-minded output, except that it restricts the choice of B to sets that are included in their own image under G.

Since each of the operations G and Cn is monotone, their composition is also monotone. Hence the definition may also be expressed thus: $out_3(G,A) = Cn(G(A^*))$ where A^* is the least superset of A that is closed under both Cn and G.

We define *basic reusable output*, written $out_4(G,A)$, as follows in the principal case that A is classically consistent:

$$out_4(G,A) = \cap\{Cn(G(V)): v(A) = 1 \text{ and } G(V) \subseteq V\}.$$

Here as before, v ranges over boolean valuations and $V = \{b: v(b) = 1\}$. In the limiting case that there is no such v, we proceed as for basic output, putting $out_4(G,A)$ to be $Cn(G(L))$ where L is the set of all boolean formulae; equal to $Cn(h(G))$ where $h(G)$ is the set of all heads of elements of G. Equivalently,

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

Clearly $out_3(G,A) \subseteq out_4(G,A) \subseteq Cn(G(L))$. The diagrams for the two notions are essentially the same. For basic reusable output, see figure 5. For the simple-minded version, replace the captions V_i by $X_i = Cn(X_i)$.

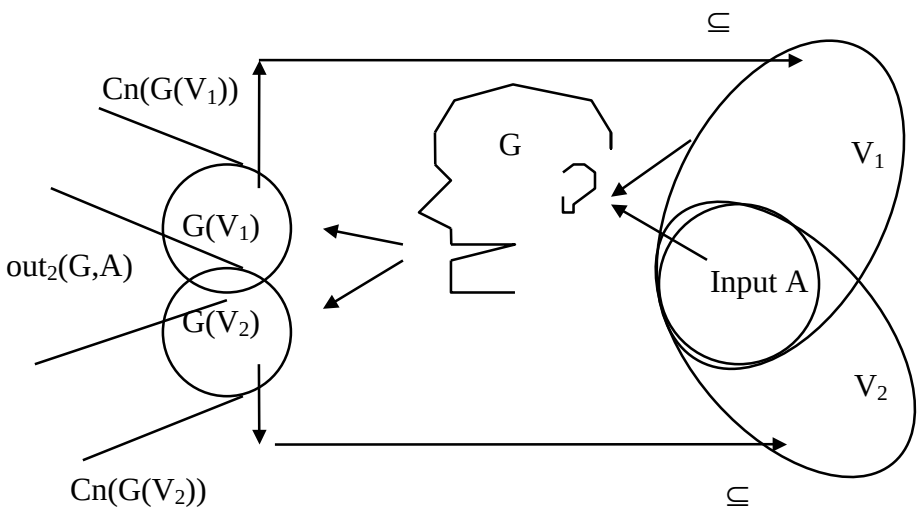


Figure 5
Basic Reusable Output: $out_4(G,A) = \cap\{Cn(G(V)): A \subseteq V \supseteq G(V)\}$

5.2. Simple-minded reusable output: properties and syntactic characterisation

As in the non-reusable case, the simple-minded reusable operation is less satisfying than the basic one, given its inability to deal intelligently with disjunctive inputs. Nevertheless, the simple-minded version has a certain interest, and we indicate some of its basic properties.

OBSERVATION 5 (cumulativity on the right). $Out_3(G,A) = out_3(G,A \cup D)$ whenever $D \subseteq out_3(G,A)$.

Proof. The left is included in the right, by monotony in the right argument (immediate from the definition). For the converse, suppose $x \notin out_3(G,A)$. Then by the definition of out_3 there is a B with $A \subseteq B = Cn(B) \supseteq G(B)$ and $x \notin Cn(G(B))$. To show $x \notin out_3(G,A \cup D)$, it suffices to show $A \cup D \subseteq B$, and so since $A \subseteq B$ and using the hypothesis $D \subseteq out_3(G,A)$, it is enough to show $out_3(G,A) \subseteq B$. But by its definition, $out_3(G,A) \subseteq Cn(G(B)) \subseteq B$ and we are done.

From cumulativity and monotony it follows immediately that simple-minded reusable output satisfies one half of idempotence on the right: $out_3(G,out_3(G,A)) \subseteq out_3(G,A \cup out_3(G,A)) = out_3(G,A)$. However, the converse half of idempotence fails.

Example 3. Put $G = \{(a,x)\}$ and $A = \{a\}$ where a,x are distinct elementary letters. Then $out_3(G,a) = Cn(x)$ whereas $out_3(G,out(G,a)) = out_3(G,Cn(x)) = Cn(\varnothing)$, so that the former is not included in the latter.

Thus for each G, the right projection function $out_{G,3}(A)$, defined as $out_3(G,A)$, is in some respects like a Tarski consequence operation (that is, a closure operation on sets of propositions) and in some respects different. It is monotonic and cumulative, and iterated output is included in single output; but in general it fails inclusion and the other half of idempotence.

These remarks about the right projection function of simple-minded reusable output should not be confused with the fact that all of our input/output operations, understood as taking sets G of pairs (A,x) to sets $out_i(G)$ of pairs, are quite trivially, closure operations - inclusion, monotony, and idempotence all hold.

We sketch a proof of the equivalence of its semantic and syntactic definitions of reusable simple-minded output, writing $deriv_3(G,A)$ for the latter.

OBSERVATION 6. $Out_3(G,A) = deriv_3(G,A)$.

Outline of proof. It suffices to prove the result for singleton A. The inclusion from right to left is straightforward by induction on length of derivation. The interesting clause is that for CT. Suppose that $x \in out_3(G,a)$ and $y \notin out_3(G,a)$; we need to show that $y \notin out_3(G,a\wedge x)$. From the second hypothesis, there is a B with $a \in B = Cn(B) \supseteq G(B)$ and $y \notin Cn(G(B))$. By the first hypothesis, $x \in Cn(G(B))$. But since $G(B) \subseteq Cn(B)$ we have $Cn(G(B)) \subseteq Cn(B)$ so $x \in Cn(B)$. Thus $a\wedge x \in Cn(B)$ and so $y \notin out_3(G,a\wedge x)$ as desired.

For the converse, suppose $x \notin deriv_3(G,a)$; we need to find a B with $a \in B = Cn(B) \supseteq G(B)$ and $x \notin Cn(G(B))$.

Put $B = Cn(\{a\}\cup deriv_3(G,a))$. Clearly $a \in B = Cn(B)$. To show $G(B) \subseteq B$, suppose $y \in G(B)$. Then there is a $b \in B$ with $(b,y) \in G$. We need to show $y \in B$, i.e. $deriv_3(G,a) \vdash a\rightarrow y$. But since $b \in B$ we have $deriv_3(G,a) \vdash a\rightarrow b$ so since $deriv_3(G,a)$ is closed under classical consequence (by the rules AND,WO and the compactness of classical consequence) we have $a\rightarrow b \in deriv_3(G,a)$, i.e. $(a, a\rightarrow b) \in deriv_3(G)$. But since $(b,y) \in G$ we also have $(b,y) \in deriv_3(G)$ so by SI, $(a\wedge b, y) \in deriv_3(G)$, so by CT, $(a,y) \in deriv_3(G)$, i.e. $y \in deriv_3(G,a)$ so by WO, $a\rightarrow y \in deriv_3(G,a)$ so $deriv_3(G,a) \vdash a\rightarrow y$ as desired.

It remains to check that $x \notin Cn(G(B))$, i.e. $x \notin Cn(G(Cn(\{a\}\cup deriv_3(G,a))))$ = $out_1(G, \{a\}\cup deriv_3(G,a)) = deriv_1(G, \{a\}\cup deriv_3(G,a))$ using the completeness theorem for simple-minded output (Observation 1). Suppose the contrary. Then, using SI, there are $x_1,\ldots,x_n \in deriv_3(G,a)$ with $x \in deriv_1(G, a\wedge x_1\wedge\ldots\wedge x_n)$ i.e. $(a\wedge x_1\wedge\ldots\wedge x_n, x) \in deriv_1(G) \subseteq deriv_3(G)$. But since each $x_i \in deriv_3(G,a)$, i.e. $(a,x_i) \in deriv_3(G)$ we have by AND and WO that $(a, x_1\wedge\ldots\wedge x_n) \in deriv_3(G)$. Hence by CT, $(a,x) \in deriv_3(G)$ i.e. $x \in deriv_3(G,a)$ contradicting our initial supposition.

5.3. Basic reusable output: first properties

We now focus on basic reusable output, better motivated than its simple-minded counterpart and also more interesting formally. To lighten terminology, from now on we refer to it simply as *reusable output*. We show in section 5.4 that $out_4(G,A) = deriv_4(G,A)$, where the latter is defined by the rules for basic output (SI,AND,WO,OR) plus CT. But before doing so we draw attention to some properties of the semantic construction.

Reusable output may equivalently be defined in the following manner, which is rather less intuitive, but establishes a link with basic output and simplifies proofs.

OBSERVATION 7. $Out_4(G,A) = \cap\{Cn(G(V)): A\cup m(G) \subseteq V, V \text{ complete}\}$.

Proof. It suffices to show that for any complete set V, we have $G(V) \subseteq V$ iff $m(G) \subseteq V$, where $m(G)$ is the materialisation of G, that is, the set of all formulae $b\rightarrow y$ with $(b,y) \in G$.

In one direction, suppose $m(G) \subseteq V$ and let $y \in G(V)$; we need to show that $y \in V$. Since $y \in G(V)$ there is a $b \in V$ with $(b,y) \in G$, so $b\rightarrow y \in V$ and so since V is complete, $y \in V$ as desired.

Conversely, suppose $G(V) \subseteq V$ and suppose $b\rightarrow y \in m(G)$; we need to show $b\rightarrow y \in V$. Suppose $b \in V$; since V is complete, it suffices to show that $y \in V$. But since $b\rightarrow y \in m(G)$ we have $(b,y) \in G$ so since $b \in V$ we have $y \in G(V) \subseteq V$ and we are done.

This observation immediately allows us to express reusable basic output in terms of its non-reusable counterpart, a fact that will be useful later.

COROLLARY TO OBSERVATION 7. $Out_4(G,A) = out_2(G,A\cup m(G))$.

It also permits a simplification of Figure 5: drop the backward-reaching lines with their inclusion signs, and alongside the input circle insert a circle for $m(G)$, also included within the V_i ellipses.

First, we note that although $out_1(G,A) \subseteq \{out_2(G,A), out_3(G,A)\} \subseteq out_4(G,A) \subseteq Cn(A\cup m(G))$, still $out_4(G,A) \neq Cn(A\cup m(G))$; in particular, inputs are still not in general outputs, and contraposition still fails, as Example 1 continues to show. Nevertheless, contraposition plays a curious 'ghostly' role for reusable basic output.

Example 4 (ghost contraposition). Put $G = \{(\neg x,\neg a),(a\wedge x,y)\}$. On the one hand, $x \notin out_4(G,a)$ since $x \notin Cn(G(L)) = Cn(h(G)) = Cn(\neg a,y)$. On the other hand, $y \in out_4(G,a)$, since $y \in Cn(G(L))$ and also for every valuation v satisfying $\{a\}\cup m(G)$, $v(x) = 1$, so $y \in G(V)$.

Expressed more generally, this example tells us that $y \in out_4(G,a)$ whenever $y \in out_4(G, a\wedge x)$ and $\neg a \in out_4(G,\neg x)$. In other words, for basic reusable output we have the rule:

GC: From $(\neg x,\neg a)$, $(a\wedge x,y)$ to (a,y).

Intuitively: although we cannot contrapose the premise $(\neg x,\neg a)$, we can 'use' the contraposition for an application of cumulative transitivity. This can be verified directly, or from the following principle of *input sufficiency*:

OBSERVATION 8 (input sufficiency). Whenever $\{a\}\cup m(G) \vdash x$, then if $y \in out_4(G, a\wedge x)$ then $y \in out_4(G,a)$. More generally, whenever $A\cup m(G) \vdash X$, then if $y \in out_4(G, A\cup X)$ then $y \in out_4(G,A)$.

Proof. Immediate from Observation 7, for if $A\cup m(G) \vdash X$ and $A\cup m(G) \subseteq V$ where V is a complete set, then $A\cup X\cup m(G) \subseteq V$.

This is a powerful principle, with a number of consequences. Expressed syntactically, it is the rule:

IS: From $(a\wedge x,y)$ to (a,y) whenever $\{a\}\cup m(G) \vdash x$.

This implies ghost contraposition, for $\neg a \in out(G,\neg x)$ implies $\{\neg x\}\cup m(G) \vdash \neg a$ so that $\{a\}\cup m(G) \vdash x$. Again, since $x \in out(G,a)$ implies $\{a\}\cup m(G) \vdash x$, input sufficiency also implies CT, which we recall authorises passage from (a,x), $(a\wedge x,y)$ to (a,y).

Essentially the same property may be expressed as follows: for reusable output we may add to the input the materialisations of some or all of the generators, without changing the output.

OBSERVATION 9 (shadow input). $Out_4(G,A) = out_4(G, A\cup m(G'))$ whenever $G' \subseteq G$.

Proof. Immediate from Observation 7, since $A\cup m(G) = A\cup m(G')\cup m(G)$ whenever $G' \subseteq G$. It may also be seen as the case of Observation 8 in which $X = A\cup m(G')$.

From Observation 9 we may say that for reusable output, generators are in a certain sense stronger than inputs. But only in a limited sense: we can *copy* from generators to inputs without altering output, but if we *transfer* from generators to inputs then we may in general lose and gain output, as can be shown by trivial examples. Simple examples also show that copying from inputs to generators may change output.

Finally, we note that basic reusable output is cumulative and satisfies half of idempotence (iterated output included in single output). The proof is the same as for simple-minded output (Observation 5). However, these properties fail for plain simple-minded and basic output (i.e. without reusability). This is as one would expect: cumulativity of the output operation is closely associated on the syntactic level with the rule CT, and on the semantic level with reusability.

5.4. Basic reusable output: syntactic characterisation

We now show that $out_4(G,A) = deriv_4(G,A)$, where the latter is defined by the rules for basic output (SI,AND,WO,OR) plus CT.

OBSERVATION 10 (soundness). Deriv$_4(G,A) \subseteq$ *out*$_4(G,A)$.

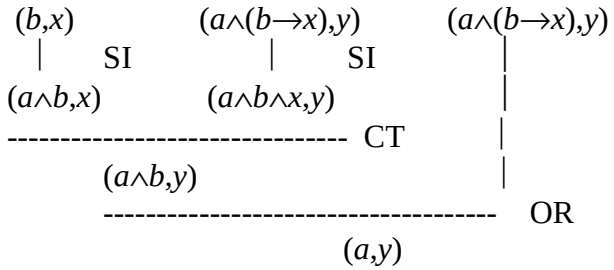
Outline of proof. We need only add to the verification of the corresponding result for basic output (Observation 2, first part of proof) a verification of the rule CT. That verification follows the same pattern as in the soundness proof for simple-minded reusable output (Observation 6).

OBSERVATION 11 (completeness). Out$_4(G,A) \subseteq$ *deriv*$_4(G,A)$.

Proof. By the Corollary to Observation 7 we have *out*$_4(G,A)$ = *out*$_2(G,A \cup m(G))$ = *deriv*$_2(G,A \cup m(G))$ by Observation 2, so it suffices to show *deriv*$_2(G,A \cup m(G)) \subseteq$ *deriv*$_4(G,A)$. Hence, we need only show that the shadow input property, already noted for the semantic operation *out*$_4$ (Observation 9), also holds for the syntactic one *deriv*$_4$. We do this in two steps: first, we prove the property for singleton input with singleton generating set, and then show that it follows in the general form.

LEMMA 11a. If $(b,x) \in G$ then *deriv*$_4(G,a \wedge (b \rightarrow x)) \subseteq$ *deriv*$_4(G,a)$.

Proof. Let $(b,x) \in G$ and suppose $y \in$ *deriv*$_4(G,a \wedge (b \rightarrow x))$; we want to show that $y \in$ *deriv*$_4(G,a)$). The desired derivation can be displayed as a tree diagram, as follows:



LEMMA 11b. *Deriv*$_4(G,A \cup m(G)) \subseteq$ *deriv*$_4(G,A)$.

Proof. Suppose $y \in$ *deriv*$_4(G,A \cup m(G))$. Clearly the operation *deriv*$_4$ is monotonic and compact on left and right. By definition, there is a conjunction a of formulae in A, and a conjunction $g = \wedge(b_i \rightarrow x_i)$ of formulae in $m(G)$, such that $y \in$ *deriv*$_4(G,a \wedge g)$. Applying Lemma 11a finitely many times according to the number of conjuncts in g, we have $y \in$ *deriv*$_4(G,a)$ so by definition $y \in$ *deriv*$_4(G,A)$. This completes the proof of the Lemma and of Observation 11.

The above proof of completeness makes use of the reduction of reusable basic output to plain basic output, in the Corollary to Observation 7. If one prefers to argue from first principles, one can re-run the same maximality construction as in the proof of Observation 2, but ensuring that $A \cup m(G) \subseteq A'$. For this it suffices to show that whenever $x \notin$ *deriv*$_4(G,A)$ then $x \notin$ *deriv*$_4(G,A \cup m(G))$, i.e. the same shadow input property *deriv*$_4(G,A \cup m(G)) \subseteq$ *deriv*$_4(G,A)$ proven as Lemma 11b above.

OBSERVATION 12 (semantic characterisation). Out$_4(G,A)$ = *deriv*$_4(G,A)$.

Proof. Immediate from Observations 10, 11. As a corollary, we may note that since $deriv_4$ is compact on both left and right, out_4 is too.

5.5. Relabeling and modal formulations

Like basic output, its reusable extension can be characterised by means of relabeling, and also in modal terms.

OBSERVATION 13. $x \in out_4(G,A)$ iff $x \in Cn(G(L))$ and $G^* \cup A \cup m(G) \vdash x^*$.

Proof. Immediate from the reduction of basic reusable output to out_2 in the Corollary to Observation 7, i.e. $out_4(G,A) = out_2(G,A \cup m(G))$, together with Observation 3.

OBSERVATION 14. $x \in out_4(G,A)$ iff $x \in Cn(G(L))$ and $G \cup A \cup m(G) \vdash_{\mathbf{S}} x$, for any modal logic **S** with $\mathbf{K_0} \subseteq \mathbf{S} \subseteq \mathbf{K45}$.

Proof: Immediate from the same reduction, with Observation 4.

A more interesting modal reduction gets rid of the 'additional premise' $m(G)$ in favour of the additional modal axiom $a \rightarrow a$, known as T.

OBSERVATION 15. $x \in out_4(G,A)$ iff $x \in Cn(G(L))$ and $G \cup A \vdash_{\mathbf{S}} x$, for any modal logic **S** with $\mathbf{K_0T} \subseteq \mathbf{S} \subseteq \mathbf{KT45}$.

Outline of proof. Since all systems from $\mathbf{K_0T}$ to **KT45** agree on first-degree formulae, we need only prove the observation for **KT**. The argument follows the same lines as for Observation 4, with the following additions and modifications.

From left to right, we need to show that the modal translation satisfies the rule CT. This amounts to showing that for any formula g, if $(g \wedge a) \rightarrow x$ and $(g \wedge a \wedge x) \rightarrow y$ are in **KT** then so is $(g \wedge a) \rightarrow y$. But this is immediate given the availability of $x \rightarrow x$ in **KT**.

From right to left, we suppose $x \notin out_4(G,A)$. As before, it follows from the definition of the output operation that there is a valuation v of boolean formulae with $v(A) = 1$ and $x \notin Cn(G(V))$, and this time we also have $G(V) \subseteq V$. Fix one such v, and define the relational model (M,R,φ) as before, but with a modified relation R. For the chosen valuation v put $(v,w) \in R$ iff for every $(b,y) \in G$, if $v(b) = 1$ then $w(y) = 1$; for every valuation $u \neq v$, put $(u,u) \in R$. Note that when $v(b) = 1$ and $(b,y) \in G$ then $y \in G(V) \subseteq V$ so that $v(y) = 1$; this shows $(v,v) \in R$. Combining this with the other part of the definition of R, we have its reflexivity, so that the model validates the modal system **KT**. To complete the proof, it suffices to check that $\varphi(v, G \cup A) = 1$ while $\varphi(v, x) = 0$. This is done exactly as in the proof of Observation 4.

We note in passing that in modal logics satisfying the modal axiom T, G implies $m(G)$, so that given observations 14 and 15, we can also push the upper bound of the former up to **KT45**.

As far as the authors are aware, it is not possible to characterise the system of simple-minded output (with or without reusability) by relabeling or modal logic in a straightforward way. The OR rule appears to be needed, so that we can work with complete sets.

6. ACCEPTING INPUTS AS OUTPUTS

What happens if we strengthen the logic of some kind of output by accepting inputs as outputs? Syntactically, add the rule: From no premises to (y,y). Evidently, such a rule can always be applied first, so the semantic counterpart amplifies $out_i(G,A)$ to $out_i(G \cup I,A)$ where $I = \{(y,y)$: y a formula$\}$. Of these, basic reusable output plus identity collapses into classical consequence.

OBSERVATION 16: $out_4(G \cup I,A) = Cn(A \cup m(G))$.

Proof. Given Observation 11, the left in right inclusion is a trivial induction. For the converse, write G^+ for $G \cup I$, and suppose $x \notin out_4(G^+,A)$. Then by Observation 7, there is a complete set V with $A \cup m(G^+) \subseteq V$ and $x \notin Cn(G^+(V))$. Clearly $V \subseteq I(V) \subseteq G^+(V) \subseteq Cn(G^+(V))$, so $x \notin V$, so that the complete set V is a maxiconsistent set, corresponding to a classical valuation v, with $v(x) = 0$. Since also $A \cup m(G) \subseteq A \cup m(G^+) \subseteq V$, we have $v(A \cup m(G)) = 1$. Putting these together, $x \notin Cn(A \cup m(G))$. □

Alternatively, one may re-run the second argument for Observation 11, observing that since $x \notin out_4(G^+,A')$ (defined as in that proof) and $I \subseteq G^+$, we have $x \notin A'$ so that $v(x) = 0$. Since $A \cup m(G^+) \subseteq A' = V$ we also have $v(A \cup m(G)) = 1$.

Simple-minded reusable output plus identity does not collapse into classical logic, but may be simplified.

OBSERVATION 17: $out_3(G \cup I,A) = \cap\{B: A \subseteq B = Cn(B) \supseteq G(B)\}$.

Proof. By the definition of out_3 it suffices to check that whenever $B = Cn(B) \supseteq G(B)$ we have $Cn(G^+(B)) = B$. Left in right: if $G(B) \subseteq B$ then since also $I(B) \subseteq B$ we have $G^+(B) \subseteq B$ so $Cn(G^+(B)) \subseteq Cn(B) = B$ by hypothesis. Right in left: since $I \subseteq G^+$ we have $B \subseteq G^+(B) \subseteq Cn(G^+(B))$. □

Note that this verification makes essential use of reusability, i.e. that $G(B) \subseteq B$, and of identity, i.e. that the generating set includes I, so that the argument does not apply to weaker kinds of output.

From our perspective, operations that accept all inputs as outputs are a limiting case of 'logically assisted transformations'. However, Observation 17 draws attention to an interesting connection with a construction underlying normal default logic. Specifically: Reiter's default logic, stripped of its consistency constraint, is the same as simple-minded reusable output with identity. To see this, take the quasi-inductive definition of an extension of a normal default system as given in (Reiter 1980, theorem 2.1) or (Makinson 1994, section 3.2), and take out the consistency constraint. This puts $ext(G,a) = \cup\{E_i: 0 \leq i < \omega\}$ where $E_1 = \{a\}$ and $E_{i+1} = Cn(E_i) \cup G(E_i)$. It is

easy to check that $ext(G,a) = \cap\{B: a \in B = Cn(B) \supseteq G(B)\}$, so by Observation 17, $ext(G,a) = out_3(G \cup I,a)$.

In (Makinson and van der Torre, to appear) we show how normal default logic, with its consistency constraint, is a special case of constrained input/output logic.

7. REVERSIBILITY OF RULES IN A DERIVATION

We finally consider briefly some questions arising for the syntactic formulations of the four input/output operations: reversibility of rules (this section) and 'universality' of certain orders of derivation of output (following section).

Note that all four input/output operations satisfy replacement of input, and of output, by classically equivalent propositions. That is, if $(a,x) \in out(G)$ then $(a',x') \in out(G)$ whenever $Cn(a) = Cn(a')$ and $Cn(x) = Cn(x')$. From this point on, we treat replacement of logically equivalent propositions as a 'silent rule', that may be applied at any step without explicit justification.

With this understanding, the order of application of two derivation rules is often 'reversible'. In some cases, we may simply permute the application of two successive rules, independently of the choice of the formulae to which they are applied. For example, any application of AND followed by SI may be replaced by one in which SI is followed by AND. In other cases, the order may be reversed, but with additional (and prior) use of a third rule - often SI and in one instance WO. Finally, there are some configurations for which no transformation appears to be available.

Observation 18 displays in a table the transformations that the authors have noted to be possible. The table should be read as follows:

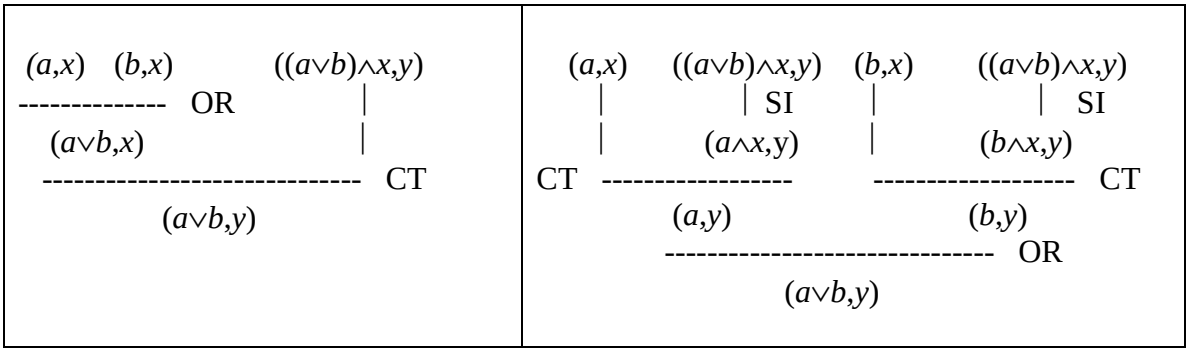
- An entry in the cell determined by the row for rule R and the column for R' tells us to what extent the sequence R,R' may be reversed to R',R.
- In an application of the asymmetric rule CT, taking us from (a,x) and $(a \wedge x,y)$ to (a,y), we call (a,x) the 'minor' premise and $(a \wedge x,y)$ the 'major' premise. In the column for CT, the left (resp. right) sub-column represents the case where the output of the previous rule feeds in as the minor (resp. major) premise of the rule CT.
- The entry ✔ indicates that simple permutation is possible.
- When only a more complex reversal is known to be possible, it is written explicitly. Thus for example in the cell for CT,AND we have written SI,AND,CT to indicate that the former order may be transformed into the latter.
- The entry *none?* means that no transformation is known to the authors.
- The empty spaces in the diagonal mean that the question does not arise there.

OBSERVATION 18 (reversibility of rules).

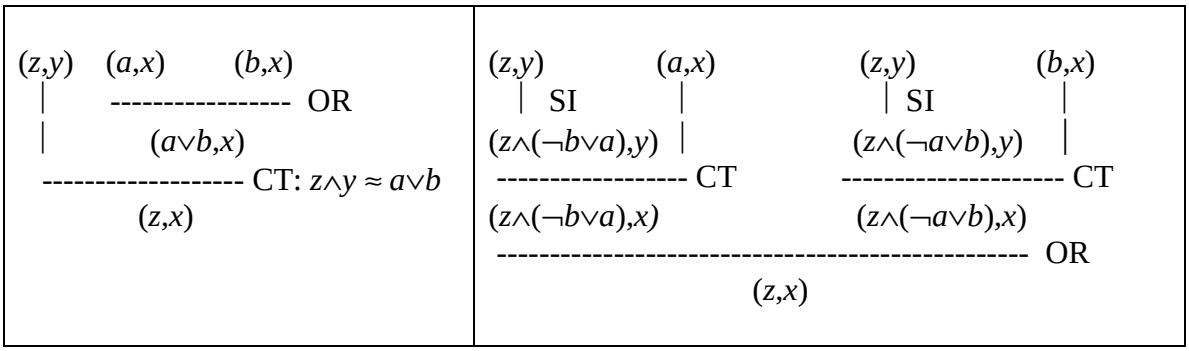
	SI	WO	CT		AND	OR
SI		✔	none?	none?	none?	✔
WO	✔		SI,CT	✔	✔	none?
CT	✔	✔			SI,AND,CT	none?
AND	✔	✔	SI,CT	✔		WO,OR,AND
OR	✔	✔	SI,CT,OR	SI,CT,OR	SI,AND,OR	

Not to overburden the paper, we omit the verifications of the reversals in the table, giving only the least immediate among them as an example. This is the transformation OR,CT $\Rightarrow$ SI,CT,OR, where the left hand configuration takes two forms according as the conclusion of OR feeds in as 'minor' or 'major' premise of the non-symmetric two-premise rule CT.

OR,CT (Case 1) $\Rightarrow$ SI,CT,OR



OR,CT (Case 2) ⇒ SI,CT,OR



Here ≈ stands for classical equivalence. In the second case, the given application of CT (on the left) is allowable iff z∧*y* ≈ *a*∨*b*, in which case *z*∧(¬*b*∨*a*)∧*y* ≈ *a* and *z*∧(¬*a*∨*b*)∧*y* ≈ *b* so that we can apply CT as indicated on the right.

8. UNIVERSAL ORDERS OF DERIVATION

Consider any system with *n* derivation rules (e.g. basic output with its four rules SI, AND, WO, OR). We say that a derivation respects an order $R_1,...,R_n$ of those rules iff a rule R_j is never applied in it before (i.e. leafwards of) a rule R_i for $i < j$. In other words, rules may be skipped or repeated (and moreover, as already mentioned earlier, it is understood that classically equivalent formulae may replace each other whenever desired), but the rules must never be applied contrary to the indicated order. Of course, many derivations do not respect any order at all; in particular, if an application of *R* is made before one of a distinct rule *R′*, but also an application of *R′* is made before one of *R*, then no order is respected in the derivation.

We say that an order is *universal* (for a given set of rules defining an input/output operation) iff whenever (*a*,*x*) ∈ *out*(*G*) then there is a derivation of (*a*,*x*) from *G* respecting that order. The question naturally arises: are there any universal orders? Repeated application of Observation 18 tells us that there are several.

OBSERVATION 19.

(a) For simple-minded output, with the rules SI, AND, WO, there are (at least) three universal orders of derivation: SI,AND,WO, and (SI,WO),AND.

(b) For basic output, with the rules SI, AND, WO, OR, there are (at least) six universal orders: SI,AND,WO, OR, and (SI,WO),(AND,OR) and WO,OR,SI,AND.

(c) For simple-minded reusable output, with the rules SI, AND, WO, CT, there are (at least) eight universal orders: SI,(WO,CT,AND) and WO,SI,(CT,AND).

(d) For reusable output, with the rules SI, AND, WO, OR, CT, there are (at least) eleven universal orders: SI,(WO,CT,AND),OR and SI,(WO,CT),OR,AND and WO,SI,(CT,AND),OR and WO,SI,CT,OR,AND.

Here the parentheses indicate that every arrangement within them is counted. The first order for simple-minded output also emerged from its completeness proof (Observation 1). Of course, Observation 19 depends very much on the particular choice of rules made, not only their joint force. For the rules that we have used, we conjecture that in each case there are no universal orders of derivation other than those listed.

Remark. In Observation 18, there are just four non-reversible orders: SI,CT; SI,AND; WO,OR; CT,OR. Thus all orders listed as universal in Observation 19 satisfy the property: SI before (immediately or separated by other rules) CT, SI before AND, WO before OR, and CT before OR. More surprisingly, it can be checked by enumeration that the converse is also true: every order satisfying that property is universal. It is not clear whether this fact points to a deeper pattern.

9. OTHER SYSTEMS

One might consider strengthening, weakening, or otherwise modifying the systems studied in this paper, with either a purely formal motivation or an eye to possible applications.

For example, with an interest in defeasible conditionals, one might drop the SI rule, perhaps replacing it by a rule of replacement of equivalent input propositions. Semantically, the operations are $Cn(G(a))$ and $Cn(G(E(a)))$ for individual formulae a, although the definition for infinite sets A (section 3.2) becomes problematic.

Again, one might consider modifying certain of the rules employed. For example, we know (section 5.1) that given SI, cumulative transitivity CT implies transitivity T, but not conversely. What happens if in the system of simple-minded reusable output, say, we replace CT by T? Given SI and AND, it is easy to show that T is equivalent to the following principle of 'ghost cumulative transitivity' GCT, which the authors have not seen in the literature: From (p,a), (a,b), $(a \wedge b,c)$ to (p,c).

We conjecture that this system may be defined as $out(G,A) = Cn(out_3(G \cup I, out_1(G,A)))$. Diagrammatically: two G boxes, one under the other, with the same ordered pairs inside. The input A comes into the first box; whose output is input to the second box. Input to the second box reappears in its output; and output of the second box is reusable in its input. The final output is closure under Cn of the second box.

Finally, one could consider adding various rules to one or more of the systems studied. For example, one could look at:

Contraposition CP:	from (a,x) to $(\neg x,\neg a)$
Dual cumulative transitivity DCT:	from $(a,x \vee y)$, (x,y) to (a,y)
Conditionalisation CND:	from (a,x) to $(t, a \rightarrow x)$.

We see these three rules of relatively minor interest, as they have little motivation in terms of the underlying vision of input/output processes outlined in the first section of this paper. Nevertheless, we note some facts about them. Observe, first, that we may add any or all of these three rules to those for basic reusable output without collapse

into classical consequence. For if $G = \varnothing$, then whenever $(a,x) \in out(G)$, where *out* is the enlarged operation, then a simple induction shows that either a is a contradiction or x is a tautology, so that in particular $(a,a) \notin out(G)$, for contingent a, whereas $a \in Cn(\{a\} \cup m(G))$. We also have the following equivalences.

OBSERVATION 20. Given CP, the rules CT and DCT are equivalent. Also, given the rules of basic output, DCT and CND are equivalent.

Outline of proof. The only verification that is not trivial is that for Basic + DCT $\Rightarrow$ CND, as follows.

$$\dfrac{\dfrac{(t,t)}{(\neg a,\ a \vee (\neg a \vee x))}\ \text{SI} \qquad \dfrac{(a,x)}{(a,\ \neg a \vee x)}\ \text{WO}}{(\neg a,\ \neg a \vee x)}\ \text{DCT} \qquad \dfrac{(a,x)}{(a,\ \neg a \vee x)}\ \text{WO}}{(t,\ a \rightarrow x)}\ \text{OR}$$

On the semantic level, it is difficult to see any input/output semantics for CP or DCT. However, in the case of the rule CND, we do have a semantics, indirectly.

OBSERVATION 21. For each of the systems out_i (i = 1,…,4), if we add the rule CND then we have a semantics like that for the source system, except that the set G is replaced by $G \cup \{(t,\ a \rightarrow x)\colon (a,x) \in G\}$. If we add both CND and the identity rule, then we replace G in the semantics by $G \cup I \cup \{(t,\ a \rightarrow x)\colon (a,x) \in G \cup I\}$.

Proof. It is easy to check that the rule CND may always be applied first in any derivation using at most SI,AND,WO,OR,CT,CND.

10. SUMMARY

The investigations in this paper are inspired by a view of logic as ‘secretarial assistant’ to an arbitrary process transforming propositional inputs into propositional outputs. Its task is to prepare the inputs, unpack the outputs, and co-ordinate the two in various ways. In this perspective, we introduced four principal input/output operations: simple-minded, basic (making intelligent use of disjunctive inputs), simple-minded reusable (in which outputs may be recycled as inputs), and basic reusable output. These are doubled by corresponding systems in which inputs reappear among the outputs. The systems are defined semantically, and are characterised syntactically by derivation rules. We recall the four basic systems.

- *Simple-minded output*, written $out_1(G,A)$, is defined as $Cn(G(Cn(A)))$, and is characterised by the rules SI,AND,WO.
- *Basic output*, written $out_2(G,A)$, is defined as $\cap\{Cn(G(V))\colon A \subseteq V,\ V \text{ complete}\}$, where a *complete* set is one that is either maxiconsistent or equal to the set L of all formulae of the language. It is characterised by SI,AND,WO,OR.

- *Simple-minded reusable output*, written $out_3(G,A)$, is defined as $\cap\{Cn(G(B)): A \subseteq B = Cn(B) \supseteq G(B)\}$. It is characterised by SI,AND,WO,CT.
- *Basic reusable output*, written $out_4(G,A)$, is defined as $\cap\{Cn(G(V)): A \subseteq V \supseteq G(V), V \text{ complete}\}$. Equivalently, as: $\cap\{Cn(G(V)): A \cup m(G) \subseteq V, V \text{ complete}\}$. It is reducible to basic output by the equality $out_4(G,A) = out_2(G,A \cup m(G))$, and is characterised by SI,AND,WO,OR,CT.

In none of the systems are inputs automatically outputs, that is, we do not have in general $a \in out(G,a)$. Nor do the systems guarantee contraposition: we may have $x \in out(G,a)$ without $\neg a \in out(G,\neg x)$. Of the four systems, basic reusable output reveals the most subtle behaviour, for example the 'input sufficiency' and 'shadow input' properties (Observations 8 and 9).

Basic output and its reusable extension may also be characterised in terms of relabeling procedures and modal operators. The account in terms of relabeling substitutes fresh elementary letters for old ones in heads, and applies classical consequence. The modal characterisation prefixes boxes to heads, and applies any modal logic from within a broad interval.

On a syntactic level, it is shown that in a surprising number of cases, the application of rules in a derivation may be reversed (Observation 18), giving rise to certain 'universal orders' of derivation for each of the four systems studied (Observation 19).

Given that an intended area of application of input/output logic is the study of systems of conditional goals or obligations, it is natural to ask how one might introduce constraints into them, to deal with 'contrary to duty' conditions. This question is investigated systematically in a sequel (Makinson and van der Torre, to appear).

REFERENCES

Makinson, David, 1999. On a fundamental problem of deontic logic. In *Norms, Logics and Information Systems. New Studies in Deontic Logic and Computer Science*, ed. Paul McNamara and Henry Prakken. Amsterdam: IOS Press, Series: Frontiers in Artificial Intelligence and Applications, Volume 49, pp 29-53.

Makinson 1994. General patterns in nonmonotonic reasoning. In *Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 3*, ed. Gabbay, Hogger and Robinson. Oxford University Press, pages 35-110.

Makinson, David, and L. van der Torre (to appear). Constraints for input/output logics.

Reiter, R. 1980. A logic for default reasoning. *Artificial Intelligence*, 13: 81-132.

van der Torre, Leendert W.N., 1997. *Reasoning about Obligations: Defeasibility in Preference-Based Deontic Logic*. Ph.D. thesis, Erasmus University of Rotterdam. Tinbergen Institute Research Series n° 140. Thesis Publishers: Amsterdam.

van der Torre, Leendert W.N., 1998. Phased labeled logics of conditional goals. *Logics in Artificial Intelligence*. Proceedings of the Sixth European Workshop on Logics in AI (JELIA'98). Berlin: Springer, LNCS 1489, pp 92-106.

ACKNOWLEDGEMENTS

Thanks to Veronica Becher, Salem Benferhat and anonymous referees for DEON 2000 for comments on drafts. Research for this paper was begun when the second author was working at IRIT, Université Paul Sabatier, Toulouse, France, and at the Max Planck Institute for Computer Science, Saarbrücken, Germany.

David Makinson
Visiting Professor, Department of Computing
King's College London
Permanent address:
Les Etangs B2, Domaine de la Ronce
92410 Ville d'Avray, France
Email: d.makinson@unesco.org

Leendert van der Torre
Department of Artificial Intelligence
Vrije Universiteit Amsterdam
De Boelelaan 1081a, 1081 HV Amsterdam, the Netherlands
Email: torre@cs.vu.nl

To appear in *Journal of Philosophical Logic*, February 2001
Last revised: 25.05.00 (transcribing corrections from page proofs)
Wordcount: 9,174