Input/Output Logics

David Makinson and Leendert van der Torre

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 general theory of propositional input/output operations. Particular attention is given to the special case where outputs may be recycled as inputs.

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 respectively facts about your date and place of birth, and 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, in preparation) we examine the imposition of consistency 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, coordinating the two. The process as a whole is one of inference only when the central transformation is so. In general, it is one of 'logically assisted' transformation. That 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.

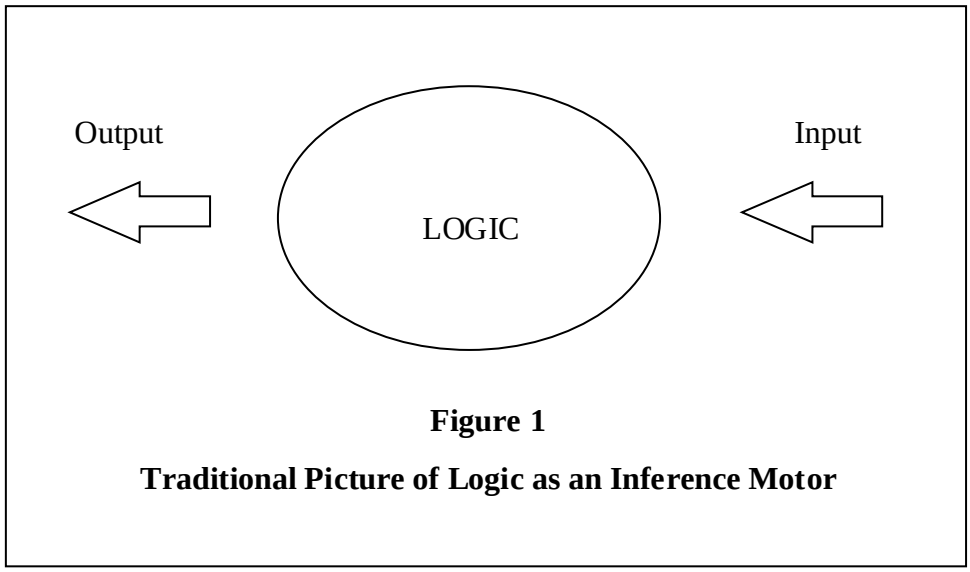


Figure 1

Traditional Picture of Logic as an Inference Motor

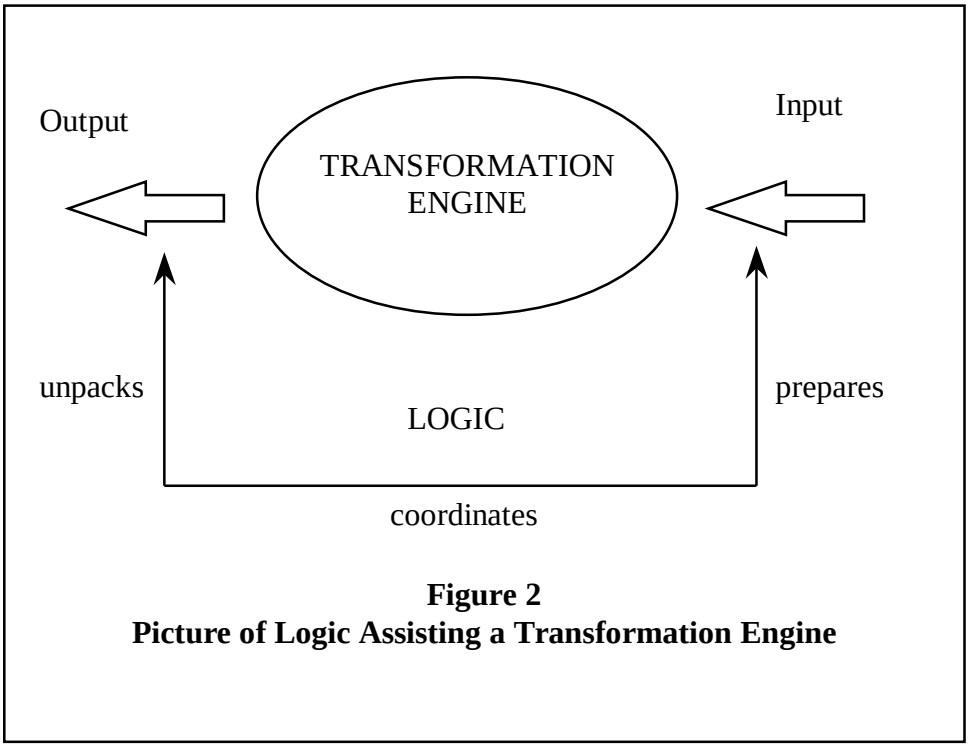


Figure 2
Picture of Logic Assisting a Transformation Engine

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 $+$ 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 $+$ 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)..

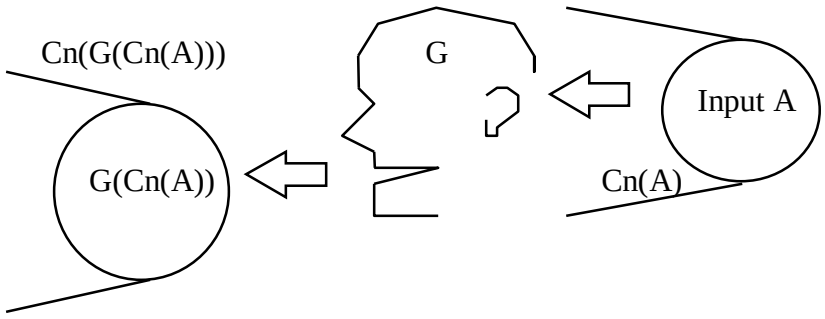


Figure 3
Simple-Minded Output: $out_1(G,A) = Cn(G(Cn(A)))$

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 $\neg a \notin out_1(G,\neg a) = Cn(G(Cn(\neg x)) = Cn(\varnothing)$.

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 for 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 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 more syntactic characterisation.

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

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

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 pair (b,x_i) and then in turn (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; equivalently $Cn(h(G))$ 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.

As is well known, $Cn(V) = V$ for any complete V, so $Cn(G(V)) = Cn(G(Cn(V))) = out_1(G,V)$, and thus $out_2(G,A) = \cap\{out_1(G,V): A \subseteq V, V \text{ complete}\}$.

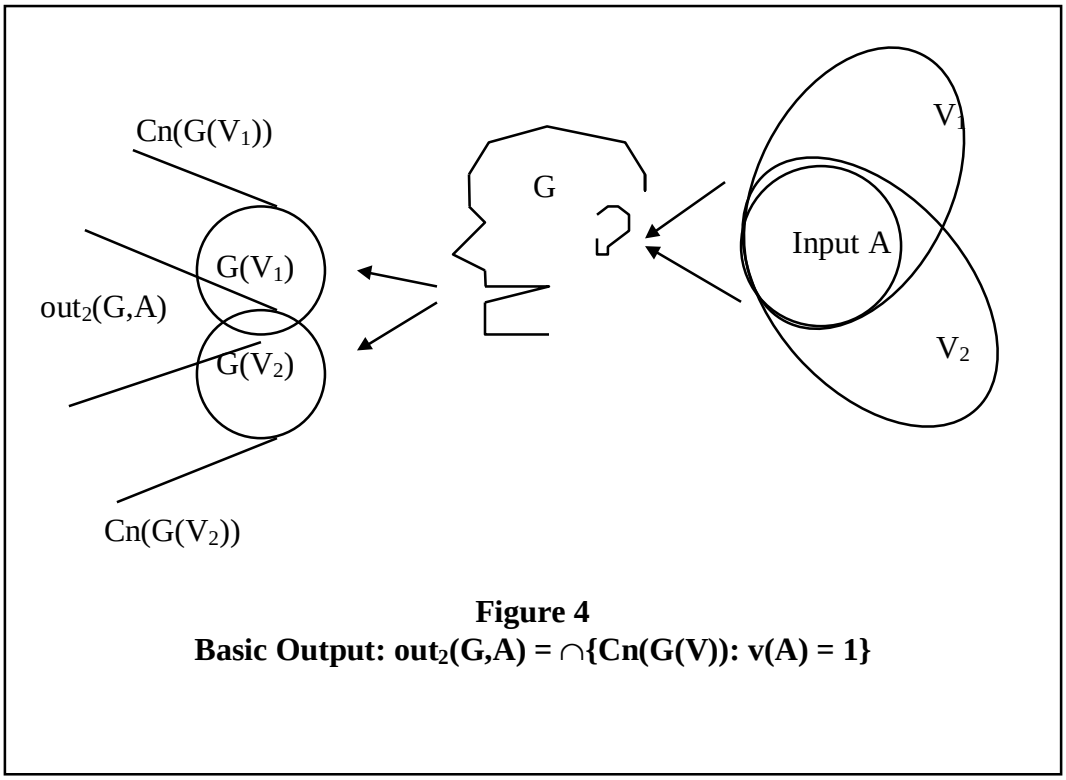


Figure 4
Basic Output: $out_2(G,A) = \cap\{Cn(G(V)): v(A) = 1\}$

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.

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 and we are done. 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 also 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 compacness 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

For the account via relabeling, introduce alongside the existing language 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 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 $x \notin Cn(G(V))$. The former is immediate from $v+w^*(A) = 1$ since A contains only unstarred letters. For the latter, it suffices to show that $w(G(V)) = 1$ while $w(x) = 0$. The latter is immediate from $v+w^*(x^*) = 0$. For the former, 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. Consider the modal propositional language formed by adding a unary box operator to the classical language, and consider the modal calculus $\mathbf{K_0}$ defined axiomatically by taking as axioms all tautologies in that language and all formulae of the form $(a \rightarrow x) \rightarrow (\Box a \rightarrow \Box x)$, and as rules passage from t to $\Box t$ for every tautology, and passage from a, $a \rightarrow x$ to x (detachment). As is well-known, this is a subsystem of the familiar system **K** (which allows passage from a to $\Box a$ for every thesis a), and it may be characterised by the set of all 'relational models with distinguished element' (M,m,R,φ) without any constraints on the relations, and with boxed formulae evaluated by φ at elements of M in the familiar manner.

Write $\Box G$ for the set of all modal formulae $b \rightarrow \Box y$ with $(b,y) \in G$, and $Z +_{\mathbf{S}} z$ 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 $G \cup \Box A +_{\mathbf{S}} x$, for any modal logic **S** with $\mathbf{K_0} \subseteq \mathbf{S} \subseteq \mathbf{K45}$.

Outline of Proof. We recall the well-known fact that for first-degree formulae (i.e. without iteration of the box) all systems from $\mathbf{K_0}$ to **K45** coincide, so we need only prove the observation for **K**. In the limiting case that A is 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 $G \cup \{a\} +_{\mathbf{K}} x$, 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 $w \in M$. We claim 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. Again, the elements (a,x) of G may prescribe what is acknowledged as output by one agent on the basis of input claims submitted by others - including itself. 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).

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

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

Since the intersection of any family of sets X such that $A \subseteq X = Cn(X) \supseteq G(X)$ satisfies the same condition, and since each of the operations G and Cn is monotone so that their composition is monotone, 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,

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

Recalling that for complete sets V (i.e. maxiconsistent or equal to L) we always have $V = Cn(V)$, we see that the definitions of out_3 and out_4 differ only in that for the latter we restrict attention to complete sets. 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 simply replace the letters V_i by $X_i = Cn(X_i)$.

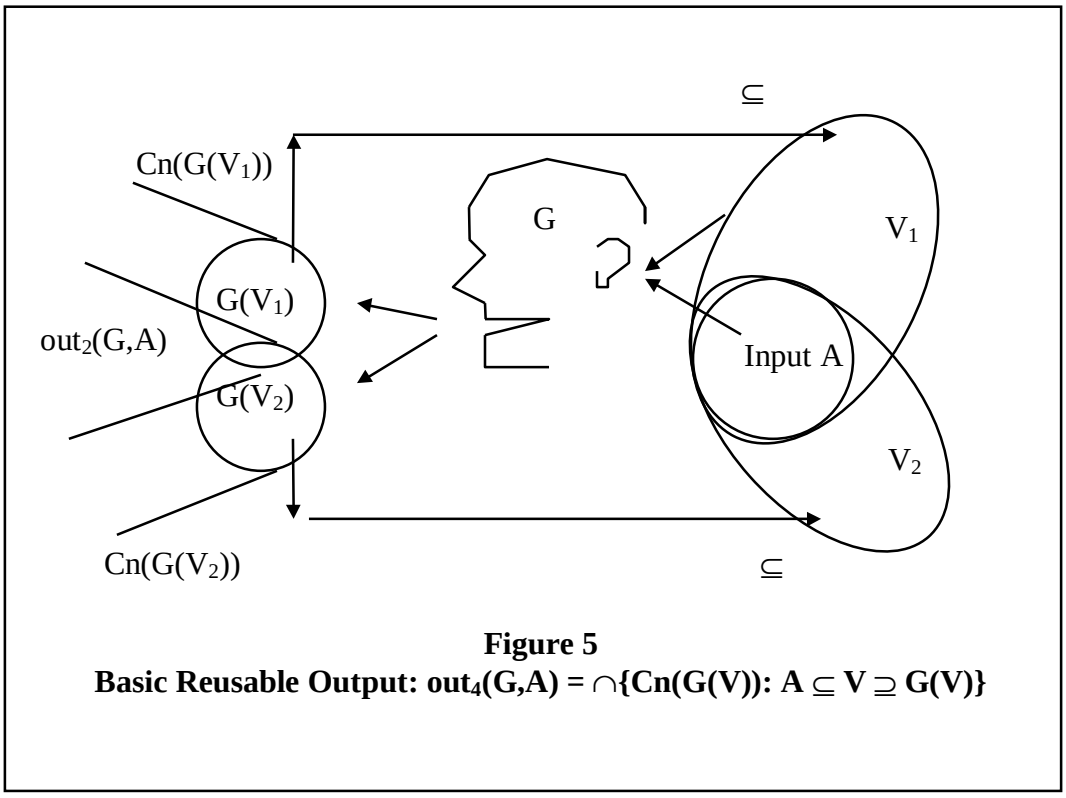


Figure 5
Basic Reusable Output: $out_4(G,A) = \cap\{Cn(G(V)): A \subseteq V \supseteq G(V)\}$

5.2. Characterisation of simple-minded reusable output

Just as basic output appears to be a more satisfactory notion than the simple-minded one, so too with their reusable counterparts. Nevertheless, here too the simple-minded

operation has a certain interest, and we sketch the proof of the equivalence of its semantic and syntactic definitions, writing $deriv_3(G,A)$ for the latter.

Observation 5. $Out_3(G,A) = deriv_3(G,A)$.

Sketch 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 an X with $a \in X = Cn(X) \supseteq G(X)$ and $y \notin Cn(G(X))$. By the first hypothesis, $x \in Cn(G(X))$. But since $G(X) \subseteq Cn(X)$ we have $Cn(G(X)) \subseteq Cn(X)$ so $x \in Cn(X)$. Thus $a \wedge x \in Cn(X)$ 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 an X with $a \in X = Cn(X) \supseteq G(X)$ and $x \notin Cn(G(X))$.

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

It remains to check that $x \notin Cn(G(X))$, 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 $(a, a \wedge x_1 \wedge \ldots \wedge x_n) \in deriv_3(G)$. 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 formally more interesting. To lighten terminology, from now on we refer to it simple as *reusable output*.

It is easy to check that for any maxiconsistent set V, we have $G(V) \subseteq V$ iff $v(m(G)) = 1$ 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 $v(m(G)) = 1$. Then whenever $y \in G(V)$ then there is a pair $(b,y) \in G$ with $b \in V$, so $v(b \rightarrow y) = 1$ and so $v(y) = 1$, and thus $y \in V$ as desired. Conversely, suppose $G(V) \subseteq V$ and suppose $(b,y) \in G$; we need to show $v(b \rightarrow y) = 1$. Suppose $v(b) = 1$; then $b \in V$ and so $y \in G(V) \subseteq V$ and thus $v(y) = 1$ as required.

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

Observation 6. $Out_4(G,A) = \cap\{Cn(G(V)): v(A \cup m(G)) = 1\}$ except in the limiting case that $A \cup m(G)$ is inconsistent, in which case $out_4(G,A) = Cn(G(L)) = Cn(h(G))$. Equivalently, $out_4(G,A) = \cap\{Cn(G(V)): A \cup m(G) \subseteq V, V \text{ complete}\}$. Equivalently, $out_4(G,A) = out_2(G,A \cup m(G))$.

This observation 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.

We shall 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. But before doing so we draw attention to some properties of the semantic construction.

First, we note that although $out_1(G,A) \subseteq out_2(G,A) \subseteq out_4(G,A) \subseteq Cn(A \cup m(G))$, 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 retains a curious 'ghostly' role.

Example 3 (ghost contraposition). Put $G = \{(\neg x,\neg a),(x,y)\}$. 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 $m(G) \cup \{a\}$, $v(x) = 1$, so $y \in G(V)$. On the syntactic level, we can construct a derivation of (a,y) from G as follows.

$$
\begin{array}{lll}
(\neg x,a) & (x,y) & (x,y) \\
\quad | \ \text{SI} & \quad | \ \text{SI} & \quad | \\
(a \wedge \neg x, \neg a) & ((a \wedge \neg x \wedge \neg a, y) & \quad | \\
\multicolumn{2}{l}{\text{----------}\ \text{CT}} & \quad | \\
\multicolumn{2}{l}{(a \wedge \neg x, y)} & (a \wedge x, y) \\
\multicolumn{3}{l}{\text{----------}\ \text{OR}} \\
\multicolumn{3}{l}{(a,y)}
\end{array}
$$

In this derivation, both CT and OR play an essential role. In the middle application of SI, the body of the conclusion is a contradiction, and so implies the head of the premise.

Expressed as a general principle, ghost contraposition tells us that $y \in out_4(G,a)$ whenever $y \in out_4(G,x)$ and $\neg a \in out_4(G,\neg x)$. Intuitively: although we cannot contrapose the second premise, we can 'use' the contraposition for an application of transitivity. This can be verified directly, or seen as following from the following principle.

Observation 7. Whenever $m(G) \cup \{a\} +\ x$, then if $y \in out_4(G,x)$ then $y \in out_4(G,a)$. More generally, whenever $m(G) \cup A +\ X$, then if $y \in out_4(G,X)$ then $y \in out_4(G,A)$.

Proof. Immediate from Observation 6, for if $m(G) \cup X +\ A$ and $A \cup m(G) \subseteq V$ where V is a complete set, then $X \cup m(G) \subseteq V$.

This implies the principle of ghost contraposition, since $\neg a \in out_4(G,\neg x)$ implies $m(G) \cup \{\neg x\} +\ \neg a$ so that $m(G) \cup \{a\} +\ x$. It can be seen as a strengthened transitivity principle, since $x \in out_4(G,a)$ also implies $m(G) \cup \{a\} +\ x$. It can also be seen as a strengthening of SI, for when $a +\ x$ then immediately $m(G) \cup \{a\} +\ x$.

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

Observation 8 (shadow input). $Out_4(G,A) = out_4(G,A \cup m(G'))$ whenever $G' \subseteq G$.

Proof. Immediate from Observation 6, since $m(G) \cup A = m(G) \cup (A \cup m(G'))$ whenever $G' \subseteq G$. May also be seen as the case of Observation 7 in which $X = m(G) \cup A$, together with monotony in the right argument (immediate from the definition).

We may thus 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.

The shadow output property implies the right cumulativity of reusable output. In other words:

Observation 9 (cumulativity on the right). $Out_4(G,A) = out_4(G,A \cup X)$ whenever $X \subseteq out_4(G,A)$.

Proof. The left is included in the right, by monotony in the right argument of reusable output. For the converse, by the hypothesis, $X \subseteq out_4(G,A) \subseteq Cn(A \cup m(G))$, so by monotony $out_4(G,A \cup X) \subseteq out_4(G,A \cup Cn(A \cup m(G))) = out_4(G,A \cup m(G))) = out_4(G,A)$ where the first equality is by replacement of logically equivalent sets on the right (immediate from the definition), and the second is by Observation 8.

Reusable output satisfies one half of idempotence on the right: $out_4(G,out_4(G,A)) \subseteq out_4(G,A \cup out_4(G,A)) = out_4(G,A)$, by monotony then cumulativity. However, the converse half of idempotence fails.

Example 4. Put $G = \{(a,x)\}$ and $A = \{a\}$ where a,x are distinct elementary letters. Then $out_4(G,a) = Cn(x)$ whilst $out_4(G,out(G,a)) = out_4(G,Cn(x)) = Cn(\varnothing)$; so that $x \in out_4(G,a)$ but $x \notin out_4(G,out_4(G,a))$.

Thus for each G, the right projection function $out_4(A)$ defined as $out_4(G,A)$ is in some respects like a Tarski consequence operation (that is, a closure operation on sets of propositions): it is monotonic and cumulative, but in general fails inclusion and half of idempotence. The right projections of basic and simple-minded reusability are also monotonic, but not in general cumulative. These observations about the right projection functions should not be confused with the fact that input/output operations, understood as taking sets G of pairs (A,x) to sets $out_i(G)$ of pairs, are quite trivially, closure operations for $i = 1,2,3,4$ - inclusion, monotony, and idempotence all hold.

5.4. Soundness and completeness for reusable output

Observation 10 (soundness). $Deriv_4(G,A) \subseteq out_4(G,A)$.

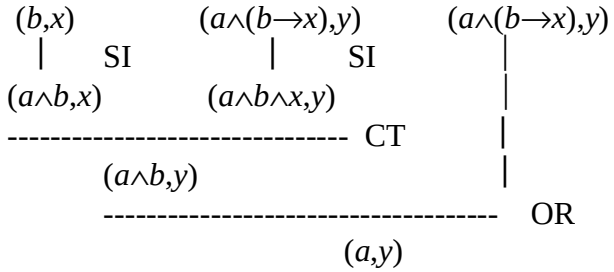
Outline of proof. We need only add to the verification of the corresponding result for basic output (Observation 2, first part of proof) a verification of the rule CT, which follows the same pattern as for simple-minded reusable output (Observation 5).

Observation 12 (completeness). $Out_4(G,A) \subseteq deriv_4(G,A)$.

Outline of proof. We need only run the same maximality construction as in the proof for basic output (Observation 2, second part of the argument), 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. that $deriv_4(G,A \cup m(G)) \subseteq deriv_4(G,A)$. That is, we need only show the shadow input property for the syntactic operation, already noted for the semantic one (Observation 8). 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 both 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 12.

Observation 13 (semantic characterisation). $Out_4(G,A) = deriv_4(G,A)$.

Immediate from Observations 10, 12. 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. We use the same notations as before.

Observation 14. $x \in out_4(G,A)$ iff $x \in Cn(G(L))$ and $G^* \cup m(G) \cup A + x^*$.

Proof. Same argument as for Observation 3, replacing A by $A \cup m(G)$throughout.

Observation 15. $x \in out_4(G,A)$ iff $x \in Cn(G(L))$ and $G \cup m(G) \cup A +_{\mathbf{S}} x$, for any modal logic **S** with $\mathbf{KT_0} \subseteq \mathbf{S} \subseteq \mathbf{KT45}$.

Outline of proof: Here we use standard notation from modal logic: $\mathbf{KT_0}$ and **KT45** are like $\mathbf{K_0}$ and **K45** with the addition of the axiom $? a \rightarrow a$. Same argument as for Observation 4, using **KT** instead of **K**, and replacing A by $A \cup m(G)$ throughout. In the completeness part, re-define the relation R by putting $(u,w) \in R$ iff either $u = v$ and for every $(b,y) \in G$, if $v(b) = 1$ then $w(y) = 1$, or $u \neq v$ and $w = u$. This relation is reflexive since $m(G) \subseteq V$. In the soundness part, the only addition to the proof is to verify that with R reflexive, the rule CT is verified, as follows. Suppose $\varphi(v, b \rightarrow ? z) = \varphi(v, b \wedge z \rightarrow ? y) = 1$; we need to show that $\varphi(v, b \rightarrow ? y) = 1$. Suppose $\varphi(v,b) = 1$. Then by the first hypothesis $\varphi(v, ? z) = 1$, so for every $w \in W$ with $(v,w) \in R$ we have $\varphi(w,z) = 1$. Since R is reflexive, in particular $(v,v) \in R$, so $\varphi(v,z) = 1$. Thus $\varphi(v, b \wedge z) = 1$, so by the second hypothesis $\varphi(v, ? y) = 1$ and we are done. ?

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 $I = \{(y,y): y \text{ a formula}\}$. Unpacking this for the four values of i, and writing G^+ for $G \cup I$, the definitions are thus as follows:

- *Simple-minded + identity*: $Cn(G^+(Cn(A)))$
- *Basic + identity*: $\cap\{Cn(G^+(V)): A \subseteq V, V \text{ complete}\}$
- *Simple-minded + reusable + identity*: $\cap\{Cn(G^+(X)): A \subseteq X = Cn(X) \supseteq G^+(X)\}$
- *Basic + reusable + identity*: $\cap\{Cn(G^+(V)): A \subseteq V \supseteq G^+(V), V \text{ complete}\}$.

The last of these collapses into classical consequence.

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

Proof. The left in right inclusion (soundness) is a trivial induction. For the converse (completeness), write G^+ for $G \cup I$, and suppose $x \notin out_4(G^+,A)$. In the case that $m(G^+) \cup A$ is inconsistent, both left and right sides equal L. Suppose that $m(G^+) \cup A$ is consistent. Then by the definition of out_4, there is a valuation v with $v(m(G^+) \cup A) = 1$ and $x \notin Cn(G^+(V))$. Since $I \subseteq G^+$ it follows that $V \subseteq G^+(V)$. Also since $v(m(G^+)) = 1$, we have $G^+(V) \subseteq V$. Thus $V = G^+(V)$. Since $x \notin Cn(G^+(V))$ we thus have $x \notin Cn(V) = V$, i.e. $v(x) = 0$. Since also $v(m(G) \cup A) = 1$, this shows $x \notin Cn(m(G) \cup A)$.

Alternatively, one may re-run the argument for Observation 12, observing that since $x \notin out_4(G^+,A)$ and $I \subseteq G^+$, we have $x \notin A'$ (defined as in that proof) so that $v(x) = 0$. Since $m(G) \cup A \subseteq A' = V$ we also have $v(m(G) \cup A) = 1$. *?*

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

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

Proof. It suffices to check that whenever $X = Cn(X) \supseteq G(X)$ we have $Cn(G^+(X)) = X$. Left in right: if $G(X) \subseteq X$ then since also $I(X) \subseteq X$ we have $G^+(X) \subseteq X$ so $Cn(G^+(X)) \subseteq Cn(X) = X$ by hypothesis. Right in left: since $I \subseteq G^+$ we have $X \subseteq G^+(X) \subseteq Cn(G^+(X))$. *?*

Note that this verification makes essential use of reusability, i.e. that $G(X) \subseteq X$, 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 $ext(G,a) = \cap\{X: a \in X = Cn(X) \supseteq G(X)\}$, so by Observation 17, $ext(G,a) = out_3(G \cup I,a)$.

7. Reversibility of rules in a derivation

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

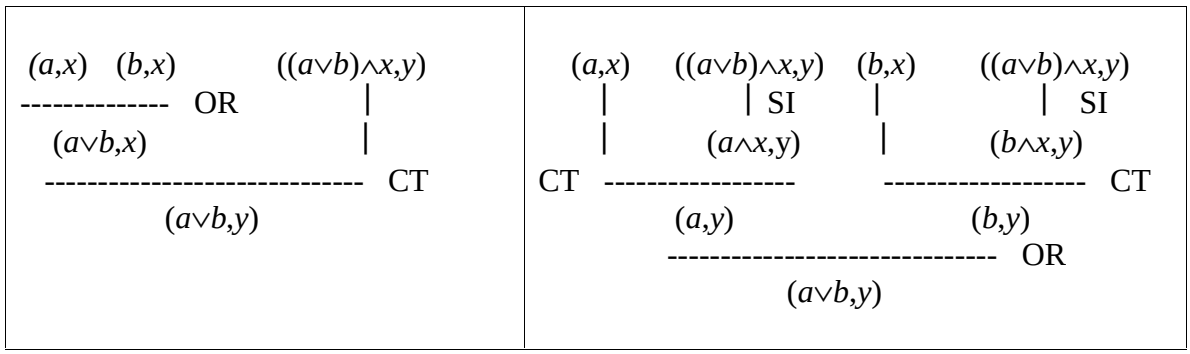
Note that all of our input/output operations (simple-minded, basic, reusable) clearly satisfy replacement of input, and of output, by classically equivalent propositions. That is, if $(a,x) \in out(G)$ then $(a',x') \in out(G)$ whenever $Cn(a) = Cn(a')$ and $Cn(x) = Cn(x')$. From this point on, we shall treat replacement of logically equivalent

propositions as a ‘silent rule’, that may be applied at any step without explicit justification.

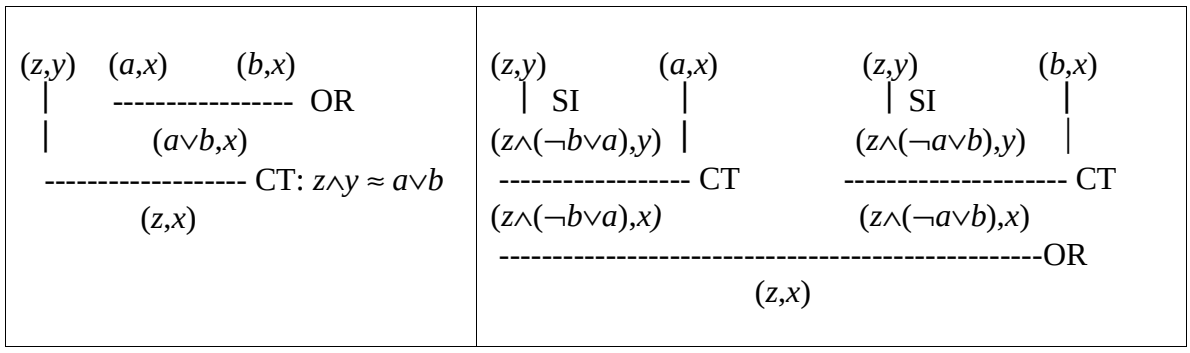
With this understanding, the order of application of two derivation rules is often ‘reversible’. In some cases, we may simply permute the application of two successive rules. For example, an application of AND followed by SI may be replaced by one in which SI is followed by AND. In other cases, the order 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.

Not to overburden the paper, we list the transformations that we have found to be possible, without detailing them in each case. As an example only, we describe in full one of the less immediate among them. 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) $\Rightarrow$ SI,CT,OR



Here ≈ stands for classical equivalence. In the second display, the given application of CT (on the left) is allowable iff $z\wedge y \approx a\vee b$, in which case $(z\wedge(\neg b\vee a),x)\wedge y \approx a$ and $(z\wedge(\neg a\vee b),x)\wedge y \approx b$ so that we can apply CT as indicated on the right.

Observation 18 displays in a table the transformations that the authors have noted to be possible. Verifications are omitted. 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*∧*x*,*y*) to (*a*,*y*), we call (*a*,*x*) the 'minor' premise and (*a*∧*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	

8. Universal orders of derivation

Consider any system with *n* rules (e.g. simple-minded output with 3 rules; basic with 4; reusable with 5). We say that a derivation respects an order $R_1,...,R_n$ of those rules iff a rule R_j is never applied 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: when a rule R is applied, then a distinct rule R', and then R again, 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 $(a,x) \in out(G)$ 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 gives us some positive answers.

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) five universal orders: SI,AND,WO, OR, and (SI,WO),(AND,OR).

(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 used, we conjecture that in each case there are no universal orders of derivation other than those listed.

9. Other rules

One may consider adding various rules to one or more of the systems studied. In particular, 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), and *conditionalisation* CZ: from (a,x) to $(t, a\rightarrow x)$. We see these systems of relatively little interest, as the rules added do not appear to be well motivated in terms of the underlying vision of a 'logically assisted transformation process' outlined in the first section of this paper. Nevertheless, for the record, we note some easily verified facts.

On the syntactic level, the rules CP, CZ, DCT are related to each other and to CT as follows. (i) Reusable + CP does not collapse into classical logic. (ii) Reusable + CP $\Rightarrow$ DCT, in other words, Basic + CT + CP $\Rightarrow$ DCT. (iii) Dually, Basic + DCT + CP $\Rightarrow$ CT. (iv) Basic + DCT $\Rightarrow$ CZ. (v) Simple-Minded + CZ $\Rightarrow$ DCT. Thus (vi) Given Basic, DCT $\Leftrightarrow$ CZ.

On the semantic level, it is difficult to see any input/output semantics for CP or DCT. However, in the case of the rule CZ, we do have a semantics, indirectly. It is easy to check that in any derivation using at most SI,AND,WO,OR,CT,CZ, the rule CZ may always be applied first. This implies that for each of our systems (simple-minded, basic, simple-minded reusable, reusable), if we add the rule CZ alone then we have a semantics like that for the source system, except that the set G is replaced by $G\cup\{(t,$

$a \rightarrow x$): $(a,x) \in G\}$. If we add both CZ and the identity rule, then we replace G in the semantics by $G \cup I \cup \{(t, a \rightarrow x)$: $(a,x) \in G \cup I\}$.

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 Leendert W.N. van der Torre (in preparation). Consistency constraints for input/output logic: a comparative review.

Reiter, R. 1980. Nonmonotonic reasoning. *Annual Reviews of Computer Science*, 2: 147-187.

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
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

Filename: input/output logics
Last revised: 06.11.99
Word count: 7,872 plus 5 figures