Journal of Philosophical Logic 30 (2001) 155-185.

DAVID MAKINSON AND LEENDERT VAN DER TORRE

CONSTRAINTS FOR INPUT/OUTPUT LOGICS

ABSTRACT. In a previous paper we developed a general theory of input/output logics. These are operations resembling inference, but where inputs need not be included among outputs, and outputs need not be reusable as inputs. In the present paper we study what happens when they are constrained to render output consistent with input. This is of interest for deontic logic, where it provides a manner of handling contrary-to-duty obligations. Our procedure is to constrain the set of generators of the input/output system, considering only the maximal subsets that do not yield output conflicting with a given input. When inputs are authorised to reappear as outputs, both maxichoice revision in the sense of Alchourrón/Makinson and the default logic of Poole emerge as special cases, and there is a close relation with Reiter default logic. However, our focus is on the general case where inputs need not be outputs. We show in what contexts the consistency of input with output may be reduced to its consistency with a truth-functional combination of components of generators, and under what conditions constrained output may be obtained by a derivation that is constrained at every step.

KEY WORDS. input/output logic, contrary-to-duty obligations, deontic logic, consistency constraints, default logics, revision.

1. BACKGROUND: INPUT/OUTPUT LOGICS

1.1. Explicit definitions

We assume familiarity with (Makinson and van der Torre, 2000), which studies unrestricted output operations. Nevertheless, for convenience, we briefly recall its central points.

We consider a Boolean context, that is, a propositional language closed under the usual truth-functional connectives. The central objects of attention are ordered pairs (a,x) of formulae, which we read forwards. Intuitively, we think of each pair (a,x) as a rule, with body a representing a possible *input*, and head x for a corresponding *output*. We call a set G of such pairs a *generating set*. The letter G also serves as a reminder of the interpretation (among others) of the pairs as conditional goals or obligations. When A is a set of formulae, we write $G(A)$ for $\{x\colon (a,x) \in G$ for some $a \in A\}$.

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

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

Here, Cn (alias $\vdash$) is classical consequence, and whenever terms like 'consequence', 'equivalence' and 'independence' are used in this paper, they are understood in their classical sense. A *complete* set is one that is either maxiconsistent or equal to the set Λ of all formulae of the language. When A is a singleton $\{a\}$, we write simply $out_i(G,a)$, and similarly for other operations defined in the paper.

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

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

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

These operations are distinct, with the exception that $out_2^+ = out_4^+$. This identity, not mentioned in (Makinson and van der Torre, 2000), may be verified as follows. The left-in-right inclusion is immediate. To show the converse, suppose $x \notin out_2^+(G,A)$. Then there is a complete set V with $A \subseteq V$ and $x \notin Cn(G^+(V))$. To prove that $x \notin out_4^+(G,A)$ we need only show that $G^+(V) \subseteq V$. But $V \subseteq G^+(V)$ so if the converse fails then $Cn(G^+(V)) = \Lambda$ contradicting $x \notin Cn(G^+(V))$.

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

1.2. Characterizations in terms of derivability

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

SI (strengthening input):	From (a,x) to (b,x) whenever $b \vdash a$
AND (conjunction of output):	From (a,x), (a,y) to $(a,x\wedge y)$

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

The reason why (t,t) is mentioned separately above is to ensure full correspondence with the explicit definition, even in limiting cases. When t is a tautology, we have $t \in out(G,a)$ even when G is empty. To derive (a,t) from G it suffices to have (t,t) and apply SI. Evidently, given SI and WO, in the above definition of $deriv(G)$ it would suffice to require $(t,t) \in deriv(G)$ for *some* tautology t.

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

ID: From no premises to (a,a)

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

1.3. Out^+/out reductions

In two cases, the operations with throughput may be reduced to their counterparts without it. In one case a reverse reduction is possible. These facts are not mentioned in (Makinson and van der Torre, 2000), so we outline the proofs.

Immediately from its definition, out_1^+ may be reduced to out_1 by the equation $out_1^+(G,A) = Cn(A \cup out_1(G,A))$.

The same identity holds for out_3, as can be shown by the following argument. The inclusion $Cn(A \cup out_3(G,A)) \subseteq out_3^+(G,A)$ is immediate, so we need only prove the converse. Given the compactness of out_3 and out_3^+ as established in (Makinson and van der Torre, 2000), it suffices to do this for singleton values of A. Thus we need only show that whenever $x \in out_3^+(G,a)$ then $a \rightarrow x \in out_3(G,a)$, i.e. whenever $(a,x) \in deriv_3^+(G)$ then $(a,a \rightarrow x) \in deriv_3(G)$. But this is easily verified by an induction on the derivation, recalling for the basis that $(t,t) \in deriv(G)$ in all our systems.

These reductions do not hold for out_2 or out_4. In the following counterexample, and all others in the paper, $a,b,x,y\ldots$ are understood to be distinct elementary letters of classical propositional logic, and thus logically independent of each other, while t is any tautology. Put $G = \{(a,x)\}$ and $A = \{\neg x\}$. Then $\neg a \in out_2^+(G,A) = out_4^+(G,A) = Cn(A \cup m(G)) = Cn(\{\neg x\} \cup \{a \rightarrow x\})$. But $\neg a \notin Cn(A \cup out_4(G,A)) \supseteq Cn(A \cup out_2(G,A))$. To see this, consider any complete set V that contains

neither a nor x, so that $G(V) = \emptyset \subseteq V \supseteq \{\neg x\} = A$ and $Cn(G(V)) = Cn(\emptyset)$, so $out_4(G,A) = Cn(\emptyset)$ and clearly $\neg a \notin Cn(\{\neg x\} \cup Cn(\emptyset))$.

In the special case of out_3 we also have a converse reduction: $out_3(G,A) = Cn(G(out_3^+(G,A))$. The left in right inclusion is immediate from the definition of out_3, since $out_3^+(G,A)$ satisfies the three conditions imposed on B in that definition. For the right-in-left inclusion, it suffices to show that $G(out_3^+(G,A))$ is included in $out_3(G,A)$, and by compactness it suffices to do this for singletons. Suppose $x \in G(out_3^+(G,a))$. Then there is a b with $(b,x) \in G$ and $b \in out_3^+(G,a)$, i.e. $(a,b) \in deriv_3^+(G)$. Hence as shown above, $(a,a\rightarrow b) \in deriv_3(G)$. Since $(b,x) \in G \subseteq deriv_3(G)$ we have by SI $(a\wedge b,x) \in deriv_3(G)$, so by CT $(a,x) \in deriv_3(G)$, i.e. $x \in out_3(G,a)$ as desired.

The reduction $out_i(G,A) = Cn(G(out_i^+(G,A))$ does not hold for any of the other output operations out_i , i = 1,2,4, as can be shown by simple examples. For out_1, out_2, put $G = \{(a,x),(x,y)\}$ and $A = \{a\}$; then the left side is $Cn(x)$ while the right side is $Cn(\{x,y\})$. For out_4, put $G = \{(a,x),(\neg a,x)\}$ and $A = \{t\}$; then the left side is $Cn(x)$ while the right side is $Cn(\emptyset)$.

2. EXCESS OUTPUT AND ITS ELIMINATION

2.1. Two kinds of excess output

Two kinds of excess are of particular interest for output: inconsistency of output *per se*, and its inconsistency with input. Since inputs are not in general authorised to reappear as outputs, these are not the same.

- Given a generating set G and input A, the output $out(G,A)$ is inconsistent iff $\perp \in Cn(out(G,A))$. Equivalently, since $out(G,A) = Cn(out(G,A))$ for all of our input/output operations, iff $\perp \in out(G,A)$.

- On the other hand, output $out(G,A)$ is inconsistent with input A iff $\perp \in Cn(out(G,A)\cup A)$. Equivalently, iff $\neg\wedge A_0 \in out(G,A)$ for some finite $A_0 \subseteq A$. When A is a singleton $\{a\}$, this comes to $\neg a \in out(G,a)$.

Here $\perp$ is the falsum (negation of a tautology). Clearly, inconsistency of output implies its inconsistency with input, but not conversely, as illustrated by the following simple example, well known from discussions of conditional norms in deontic logic.

EXAMPLE 1 (*broken promise*). Let $G = \{(t,\neg a), (a,x)\}$ where t is a tautology. To give it flesh, read a as 'you break your promise' and x as 'you apologize'. Then $out(G,a) = Cn(\{\neg a,x\})$ for each of the four principal input/output operations out_n (n = 1,2,3,4). Thus $\perp \notin out(G,a)$, i.e. output is consistent. However $\neg a \in out(G,a)$, i.e. output is inconsistent with input.

It is clear why we may wish to ensure consistency of output. But why might we also want to guarantee its consistency with input? The motivation comes from deontic logic. Suppose we are given a code G of conditional norms. Imagine that we are presented with a condition (input) that is unalterably true, and ask what obligations (output) it gives rise to. It may happen that the

condition is something that should not have been true in the first place. But that is now water under the bridge; we have to "make the best out of the sad circumstances" as (Hansson 1969) put it. We therefore abstract from the deontic status of the condition, and focus on the obligations that are consistent with its presence. In the above example, if the person has not kept a promise, then we want to know what should be done consistent with that situation. How to determine this in general terms, and if possible in formal ones, is the well-known problem of contrary-to-duty conditions.

2.2. Avoiding excess output: maxfamilies and outfamilies

Our strategy for eliminating excess output is to cut back the set of generators to just below the threshold of yielding excess. To do that, we adapt a technique that is well known in the more specific areas of belief change and nonmonotonic inference – look at the maximal non-excessive subsets.

The formal definition is general, covering as special cases both inconsistency of output and its inconsistency with input. Let G be a generating set, and let C be an arbitrary set of formulae, which we will call the 'constraint set'. For any input set A, we define $maxfamily(G,A,C)$ to be the family of all maximal $H \subseteq G$ such that $out(H,A)$ is consistent with C.

The cases $C = \emptyset$ and $C = A$ express consistency of output, and its consistency with input. In other words with $C = \emptyset$ (respectively $C = A$), $maxfamily(G,A,C)$ gathers the maximal $H \subseteq G$ such that $out(H,A)$ is consistent (respectively consistent with input A). For throughput operations $out = out_n^+$ we have $A \subseteq out(G,A)$, so that $maxfamily(G,A,\emptyset) = maxfamily(G,A,A)$. But for the operations $out = out_n$ without throughput, they are quite different.

Care should be taken when applying the definition to the throughput operations. $Maxfamily(G,A,C)$ is understood to be the family of all maximal $H \subseteq G$ such that $out_n^+(H,A) = out_n(H \cup I,A)$ is consistent with C. It is not the family of all maximal $H \subseteq G \cup I$ such that $out_n(H,A)$ is consistent with C. In other words, the set I is protected from attrition.

We define $outfamily(G,A,C)$ to be the family of outputs under input A, generated by elements of $maxfamily(G,A,C)$. In other words, $outfamily(G,A,C)$ is the family of all sets $out(H,A)$ such that H is maximal among the subsets $H' \subseteq G$ with $out(H',A)$ consistent with C.

2.3. Meets and joins of outfamilies

As one would expect from the analogous constructions in the logics of belief change and nonmonotonic inference, the definition of an outfamily gives rise to notions of full meet and full join constrained output, i.e. $\cap(outfamily(G,A,C))$ and $\cup(outfamily(G,A,C))$. Some special cases of these operations have been studied in work on conditional norms in deontic logic. For example, (Hansson and Makinson 1997) in effect give a way of constructing $\cap(outfamily(G,a,\emptyset))$ for $out = out_2$. As we shall see in Sections 5 and 6, certain truth-functional approximations to $\cup(outfamily(G,A,A))$ have also been studied in connection with contrary-to-duty conditional norms.

As in the logics of belief change and nonmonotonic reasoning, in addition to the full meet and join operations, there are partial ones, $\cap(\gamma(outfamily(G,A,C)))$ and $\cup(\gamma(outfamily(G,A,C)))$. Here γ is any selection function, with $\gamma(Y)$ being a subset of Y, non-empty if Y is. Thus the full meet and full join operations are uniquely defined, while partial meet and join are schemas whose value depends on γ. The meet operations are closed under classical consequence since the intersected sets $out(H,A)$ are all closed, but in general the join operations are not closed under classical consequence.

In this paper we focus on the outfamilies themselves, without investigating systematically their meets and joins. Nevertheless, it will be useful when we consider constrained derivations in Sections 6 and 7, to note that full join constrained output may be characterized more directly, without mention of maximality.

OBSERVATION 1. $x \in \cup(outfamily(G,A,C))$ iff there is a $H \subseteq G$ such that $x \in out(H,A)$ and $out(H,A)$ is consistent with C.

Proof. Left to right is immediate from the definition of $\cup(outfamily(G,A,C))$. For the converse, suppose there is a $H \subseteq G$ such that $out(H,A)$ contains x and is also consistent with C. Now each of our seven input/output operations is compact in its left argument G; this is immediate from their characterizations by derivation rules in Section 1.2. Hence by Zorn's Lemma, there is a maximal H' with $H \subseteq H' \subseteq G$ such that $out(H',A)$ is consistent with C. Thus $H' \in maxfamily(G,A,C)$. By the monotony of the unrestricted output operation $out(G,A)$ in its left argument, since $x \in out(H,A)$ we have $x \in out(H',A)$. Thus $x \in \cup(outfamily(G,A,C))$.

In Appendix #1 we note the monotony/antitony properties of the full join and meet operations, with respect to arguments G,A,C.

3. SOME EXAMPLES FROM DEONTIC LOGIC

We give some examples that are familiar from discussions of contrary-to-duty norms in deontic logic. We calculate separately for $C = \varnothing$ and $C = A$. In Examples 3.1 and 3.2 the calculation holds indifferently for the four principal input/output operations not authorising throughput. In Example 3.3 we calculate for out_3 and out_4.

EXAMPLE 3.1. We return to the *broken promise* (Example 1) and reconsider it in the light of our formal definitions. Recall that $G = \{(t,\neg a), (a,x)\}$ where t is a tautology, a is 'you break your promise' and x is 'you apologize'. Then as already noted, $out(G,a) = Cn(\neg a,x)$ which is consistent. Thus on the one hand, for $C = \varnothing$:

$$maxfamily(G,a,\varnothing) = \{G\}$$
$$outfamily(G,a,\varnothing) = \{Cn(\neg a,x)\}$$
$$\cap(outfamily(G,a,\varnothing)) = \cup(outfamily(G,a,\varnothing)) = Cn(\neg a,x)$$

On the other hand, $out(G,a)$ is not consistent with the input a. There is a just one maximal subset of G whose output is consistent with input a, namely the singleton $\{(a,x)\}$. Thus for $C = \{a\}$ we have:

$$maxfamily(G,a,a) = \{\{(a,x)\}\}$$
$$outfamily(G,a,a) = \{Cn(x)\}$$
$$\cap(outfamily(G,a,a)) = \cup(outfamily(G,a,a)) = Cn(x)$$

This agrees with the intuitive assessment of the example, where the elements of G are understood as conditional obligations. Given that one has broken the promise, the obligation to apologize becomes operative, while the obligation not to violate the promise is no longer in play.

EXAMPLE 3.2. (*broken promise without apology*). Multiple levels of violation may be analysed in the same way. For example, put $G = \{(t,\neg a), (a,x), (a\wedge\neg x,y)\}$ where t,a,x are as in Example 3.1 and y is 'you are ashamed'. Consider the input $a\wedge\neg x$.

Then $out(G,a\wedge\neg x) = Cn(\neg a,x,y)$, which is consistent, so that $maxfamily(G, a\wedge\neg x, \emptyset) = \{G\}$ and $outfamily(G, a\wedge\neg x, \emptyset) = \{Cn(\neg a,x,y)\}$. On the other hand, $out(G,a\wedge\neg x)$ is inconsistent with input $a\wedge\neg x$, so that $maxfamily(G, a\wedge\neg x, a\wedge\neg x) = \{(a\wedge\neg x, y)\}$ and $outfamily(G, a\wedge\neg x, a\wedge\neg x) = \{Cn(y)\}$.

EXAMPLE 3.3. (*Möbius strip, Makinson* 1994, 1999). Put $G = \{(a,b), (b,c), (c,\neg a)\}$. For instance, a,b,c could represent 'Alice (resp. Bob,Carol) is invited to dinner'. The pair (a,b) then tells us that if Alice is invited then Bob should be, and so on. Consider a as input. Calculating for $out \in \{out_3, out_4\}$, we have $out(G,a) = Cn(b,c,\neg a)$ which is consistent. Hence:

$$maxfamily(G,a,\emptyset) = \{G\}$$
$$outfamily(G,a,\emptyset) = \{Cn(b,c,\neg a)\}$$
$$\cap(outfamily(G,a,\emptyset)) = \cup(outfamily(G,a,\emptyset)) = Cn(b,c,\neg a).$$

But $out(G,a)$ is not consistent with the input a. This time there are three maximal subsets of G whose output (under input a) is consistent with a, namely the three two-element subsets. Thus:

$$maxfamily(G,a,a) = \{\{(a,b),(b,c)\}, \{(a,b),(c,\neg a)\}, \{(b,c),(c,\neg a)\}$$
$$outfamily(G,a,a) = \{Cn(b,c), Cn(b), Cn(\emptyset)\}$$
$$\cap(outfamily(G,a,a)) = Cn(\emptyset)$$
$$\cup(outfamily(G,a,a)) = Cn(b,c).$$

The Möbius strip is a fascinating example. Since $Cn(\emptyset) \subset Cn(b) \subset Cn(b,c)$, it shows that for out_3 elements of the outfamily are not always maximal. In Appendix #2 we investigate this matter further.

4. SPECIAL CASES FOR DEFAULT LOGIC AND REVISION

We show how Poole systems and maxichoice revision may both be seen as a special case of constrained input/output logic, and that normal Reiter default systems are closely related to another one. Specifically, Poole systems and maxichoice revision correspond to constrained reusable basic throughput out_4^+, while normal Reiter defaults are closely related to constrained out_3^+. This section may be omitted without loss of continuity for the general theory, but those readers already familiar with the above systems may find the connections revealing.

OBSERVATION 4. Let (D,A,C) be a Poole default system, where D is the set of its default formulae, A its set of premises, and C its set of constraining formulae. Let $extfamily(D,A,C)$ be the family of its extensions in the sense of Poole. Then $extfamily(D,A,C) = outfamily(G,A,C)$, where $G = \{(t,x): x \in D\}$ and *outfamily* is defined using reusable basic throughput out_4^+.

Proof. Recall again from Section 1.1 that $out_4^+(G,A) = Cn(A \cup m(G))$. Note also that since $t \rightarrow x$ is classically equivalent to x, $m(G)$ is equivalent to D.

By the definition of a Poole default system – see (Poole 1988) or the exposition in (Makinson 1994, Section 3.3) – $extfamily(D,A,C)$ is the family of all sets $Cn(A \cup D')$ with D' a maximal subset of D such that $Cn(A \cup D')$ is consistent with C. By the construction of G, this is identical to the family of all sets $Cn(A \cup m(H))$ such that H is a maximal subset of G with $Cn(A \cup m(H))$ consistent with C. Since $out_4^+(H,A) = Cn(A \cup m(H))$, we conclude that $extfamily(D,A,C)$ is the family of all sets $out_4^+(H,A)$ such that H is a maximal subset of G with $out_4^+(H,A)$ consistent with C, i.e. it is $outfamily(G,A,C)$.

In the logic of belief change, the well-known partial meet revisions of (Alchourrón, Gärdenfors and Makinson 1985) have maxichoice revisions as a special case, already studied by (Alchourrón and Makinson 1982). It is well-known that maxichoice revisions may also be regarded as Poole default extensions: the maxichoice revisions $K*a$ are precisely the extensions of the Poole default system $(K,\{a\},\emptyset)$. Thus from Observation 4 (or by direct verification) we also have:

COROLLARY TO OBSERVATION 4. Let K be a belief set and a any formula. Let $revfamily(K,a)$ be the set of all maxichoice revisions of K by a. Then $revfamily(K,a) = outfamily(G,a,\emptyset) = outfamily(G,a,a)$, where $G = \{(t,x): x \in K\}$ and *outfamily* is defined using out_4^+.

Normal default systems in the sense of (Reiter 1980) are closely related to input/output logics with out_3^+ (reusable simple-minded throughput) as the underlying unconstrained operation. However, the correspondence is less complete than for Poole systems. The family of Reiter extensions forms a distinguished subset of the corresponding outfamily. Roughly speaking, Reiter extensions maximize output, while constrained outputs maximize subsets of G, whose outputs need not be maximal (cf. Appendix #2).

This may be illustrated by looking again at Example 3.3, the Möbius strip, where $G = \{(a,b), (b,c), (c,\neg a)\}$, and which we have already calculated for $out \in \{out_3, out_4\}$. Recalculating for $out = out_3^+$, we get a very similar pattern. On the one hand, there are three elements in the outfamily, forming a chain, as follows:

$out(G,a) = Cn(a,b,c,\neg a)$
$maxfamily(G,a,\emptyset) = maxfamily\ (G,a,a) = \{\{(a,b),(b,c)\}, \{(a,b),(c,\neg a)\}, \{(b,c),(c,\neg a)\}$
$outfamily(G,a,\emptyset) \ = outfamily(G,a,a) \ = \{Cn(a,b,c), Cn(a,b), Cn(a)\}$

On the other hand, if we read the elements of G as normal Reiter default rules $a;b/b$ etc., then (G,a) has a unique Reiter extension $Cn(a,b,c)$, which is the largest element of the outfamily.

The general relationship between outfamily and extfamily is given by the following observation. To simplify notation, we write $outfamily(G,A)$ indifferently for $outfamily(G,A,A)$ and $outfamily(G,A,\emptyset)$, knowing that these are identical for input/output operations out_n^+ admitting throughput. For simplicity of notation we identify a normal rule $a;x/x$ with the corresponding pair (a,x).

OBSERVATION 5. Let (G,A) be any normal Reiter default system, with G the set of its default rules and A the set of its premises. Suppose that A is consistent. Write $extfamily(G,A)$ for the family of all its extensions in the sense of Reiter. Let *outfamily* be defined using reusable simple-minded throughput out_3^+. Then:

(a) $extfamily(G,A) \subseteq outfamily(G,A)$
(b) for every $X \in outfamily(G,A)$, there is an $E \in extfamily(G,A)$ with $X \subseteq E$.

Proof. See Appendix #3.

COROLLARY TO OBSERVATION 5. Under the same conditions as Observation 5, $extfamily(G,A)$ consists of exactly the maximal elements of $outfamily(G,A)$. In brief: $extfamily(G,A) = max(outfamily(G,A))$.

Proof. Immediate from Observation 5, using the fact that no extension is properly included in any other (Reiter 1980, Theorem 2.4).

SECOND COROLLARY TO OBSERVATION 5. Under the same conditions as Observation 5, $\cup(extfamily(G,A)) = \cup(outfamily(G,A))$.

Proof. Immediate from Observation 5.

Observation 5 also implies that for $out = out_3^+$ and consistent A, we have the following 'embedding property': every element of $outfamily(G,A)$ is included in a maximal such element. However, this is a rather roundabout way to prove the property, via Reiter default systems. It also leaves several questions open. Can we drop the hypothesis of the consistency of A? Can we prove the property more generally for an arbitrary constraint set C? Can we prove it for other values of out_i?

We can prove it for out_1, out_1^+, and out_4^+ (alias out_2^+). Indeed for those operations we have a much stronger property: no element of $outfamily(G,A,C)$ is properly included in any other (Observation 3 in Appendix #2). But for the remaining operations out_2, out_3, out_4, this stronger property fails and the status of the embedding property remains open.

We end this section with a terminological warning. The term 'generating set' is used in this paper in a sense quite different from that of (Reiter 1980, Definition 2). For us, a generating set is any set of pairs (a,x) of formulae. Reiter uses the term in a quite different sense. Given a default system (G,A) and a set X of formulae, the generating set for X, in Reiter's sense, is the set of all default rules in G whose prerequisites are in X and whose justifications are consistent with X.

5. TRUTH-FUNCTIONAL REDUCTIONS OF THE INPUT/OUTPUT CONSTRAINT

5.1. Materialisations, heads, and fulfilments

We recall from Section 2.1 that a principal motivation for studying constrained input/output logics is their application to deontic contexts, and in particular to contrary-to-duty conditional norms. For this reason, from this point on we restrict attention to the requirement that output $out(G,A)$ is consistent with input A, i.e. the case that $C = A$. We call this the *input/output constraint*. It gives the sets $maxfamily(G,A,A)$ and $outfamily(G,A,A)$.

One may imagine truth-functional counterparts to this constraint. For example, one could require that input A is consistent with the set $m(G)$ of materialisations of elements of G, i.e. the set of all formulae $a \rightarrow x$ with $(a,x) \in G$. Again, we could require A to be consistent with the set $h(G)$ of heads x of elements (a,x) of G, or with the set $f(G)$ of its fulfilments $a \wedge x$. The authors have studied some of these truth-functional approximations in earlier work on conditional norms. (Van der Torre 1997, 1998) examined the fulfilment constraint as applied to derivations, first for out_3 and then for out_4. (Makinson 1999) investigated the head constraint for out_4 in the same context.

What is the relationship between the input/output constraint and its truth-functional counterparts? In one direction, we have the following simple fact.

OBSERVATION 6. Let *out* be any of our seven unrestricted input/output operations. For all $C \supseteq A$, if C is consistent with a set in the list $out(G,A)$, $m(G)$, $h(G)$, $f(G)$ then it is consistent with any set earlier in the list.

Proof. Recall from Section 1.1 that $out(G,A) \subseteq Cn(A \cup m(G))$. It follows that for all $C \supseteq A$, if C is consistent with $m(G)$ and thus also with $Cn(A \cup m(G))$, then it is consistent with $out(G,A)$. For the remainder, simply note that by classical logic $f(G) \vdash h(G) \vdash m(G)$.

In the converse direction, the situation depends on the choice of the background input/output operation, as we now show.

5.2. Truth-functional reductions

We begin by noting that the converse of Observation 6 fails for out_i when $i \in \{1,1^+,2,3,3^+\}$, even in the case that $C = A$ and even for the 'nearest' truth-functional counterpart $m(G)$. We give two examples that together cover the operations.

EXAMPLE 4.1. For $i \in \{1,1^+,2\}$, put $G = \{(t,x), (x,y)\}$ and $A = \{\neg y\}$. Then A is inconsistent with $m(G)$ and so with $h(G)$ and $f(G)$. On the other hand, A is consistent with $out_i(G,A)$, since $out_1(G,A) \subseteq out_1^+(G,A) = Cn(G^+(Cn(A))) = Cn(\{x,\neg y\})$ while $out_2(G,A) = Cn(\{x\})$, both of which are consistent with $\neg y$.

EXAMPLE 4.2. For $i \in \{1,1^+,3,3^+\}$, Put $G = \{(a,x), (\neg a,x)\}$ and $A = \{\neg x\}$. Then A is inconsistent with $m(G)$ and so with $h(G)$ and $f(G)$. On the other hand, A is consistent with $out_3^+(G,A) \supseteq out_i(G,A)$. To see this, put $B = Cn(\neg x)$. Then $A \subseteq B = Cn(B) = G^+(B) = Cn(\neg x)$ so $out_3^+(G,A) \subseteq Cn(\neg x)$ which is consistent with $\neg x$.

On the other hand, we have a converse for out_4^+ (which, we recall, coincides with out_2^+), and another one for out_4 in the case $C = A$.

OBSERVATION 7. For all $C \supseteq A$, C is consistent with $out_4^+(G,A)$ iff it is consistent with $m(G)$.

Proof. Right to left is already given by Observation 6. For left to right, recall from Section 1.1 that $out_4^+ = Cn(A \cup m(G))$ and conclude.

OBSERVATION 8. A is consistent with $out_4(G,A)$ iff it is consistent with $m(G)$.

Proof. Right to left is already given by Observation 6. For left to right, suppose A is inconsistent with $m(G)$; we need to show that it is inconsistent with $out_4(G,A)$. Let V be any complete set with $A \subseteq V \supseteq G(V)$. Then $m(G) \subseteq V$, for otherwise there is a pair $(a,x) \in G$ with $a \rightarrow x \notin V$ so $a \in V$ and $\neg x \in V$ contradicting $G(V) \subseteq V$. Since both A and $m(G)$ are included in V, $V = \Lambda$. It follows that $out_4(G,A) = Cn(G(\Lambda)) = Cn(h(G)) \supseteq Cn(m(G)) \supseteq m(G)$. Since A is inconsistent with the last, it is inconsistent with the first, and we are done.

COROLLARY TO OBSERVATIONS 7,8. For $out \in \{out_4, out_4^+\}$, $maxfamily(G,A,A)$ is the family of all maximal $H \subseteq G$ such that A is consistent with $m(H)$; thus also $outfamily(G,A,A)$ is the family of sets $out(H,A)$ for maximal $H \subseteq G$ such that A is consistent with $m(H)$.

5.3. The effect of throughput on maxfamilies

From Observations 7 and 8, together with results in Section 1.3, we obtain a further corollary, on the effect of throughput on consistency of output with input, and thus in turn on the identity of maxfamilies.

SECOND COROLLARY TO OBSERVATIONS 7,8. For $out \in \{out_1, out_3, out_4\}$, the following three conditions are equivalent: A is consistent with $out(G,A)$, A is consistent with $out^+(G,A)$, $out^+(G,A)$ is consistent.

Proof. For out_1 and out_3 this immediate from the reduction $out^+(G,A) = Cn(A \cup out(G,A))$ established for those operations in Section 1.3, recalling that $A \subseteq out^+(G,A)$. For out_4, apply Observations 7 and 8.

THIRD COROLLARY TO OBSERVATIONS 7,8. For $out \in \{out_1, out_3, out_4\}$, $maxfamily(G,A,A) = maxfamily^+(G,A,A)$.

Proof. Immediate from the Second Corollary. Notation: when *maxfamily* is determined by *out*, then $maxfamily^+$ is understood to be determined by out^+.

Note however that for the same operations out_1, out_3, out_4, we may have $outfamily(G,A,A) \neq outfamily^+(G,A,A)$ since in general $out(H,A) \neq out^+(H,A)$.

Moreover, for out_2 the identity $maxfamily(G,A,A) = maxfamily^+(G,A,A)$ fails. For a counterexample, put $G = \{(a,x), (x,\neg a)\}$. Then using characterizations in Section 1.1, $\neg a \in out_2^+(G,a) = out_4^+(G,a) = Cn(\{a\} \cup m(G))$. But $\neg a \notin out_2(G,a)$, as witnessed by any complete set containing a but not x, for then $G(V) = \{x\}$ and $\neg a \notin Cn(x)$.

6. CONSTRAINED DERIVATIONS

6.1. Definitions

Up to this point, we have considered constrained output in terms of its explicit definition, i.e. from a perspective that may be called, in a broad sense of the word, semantic. We now examine it in terms of derivations, where a number of new questions arise.

We know (Section 1.2) that each of our unconstrained input/output operations may be given a characterization in terms of derivations: $x \in out(G,a)$ iff (a,x) is derivable from G using appropriate rules. In this section we consider constraint as a requirement on the root of a derivation, relative to its leaves. In Section 7, we examine a more demanding way of applying it, as a requirement on every step of a derivation.

In Section 1.2 we sketched the notion of derivability, but leaving implicit the definitions of a rule and of a derivation. We now need to be more explicit. A *rule* r (of arity $n \geq 0$) is a subset of P^{n+1} where P is the set of all ordered pairs of formulae. When $((a_1,x_1),\ldots, (a_n,x_n), (a_{n+1},x_{n+1})) \in r$, then $(a_1,x_1),\ldots, (a_n,x_n)$ are called its premises and (a_{n+1},x_{n+1}) its conclusion.

A *derivation* of a pair (a,x) from a set G of pairs of formulae, given a set R of rules, is understood to be a tree with (a,x) at the root, each non-leaf node related to its immediate parents by the inverse of a rule in R, and each leaf node either the conclusion of a zero-premise rule in R, or an element of G, or of the form (t,t). It is understood that not all elements of G need to appear as leaves. Nor do all rules in R need to be applied in the derivation, but no others may be employed.

A pair (a,x) of formulae is said to be *derivable* from G given rule-set R, and we write $(a,x) \in deriv(G)$ or $x \in deriv(G,a)$, iff there is some derivation of (a,x) from G given R. Equivalently, recalling the formulation in Section 1.2, $deriv(G)$ is the least set that includes G, contains the pair (t,t) where t is any tautology, and is closed under the rules.

We say that a derivation Δ is constrained iff the body of the root is consistent with its own derivability set. To be precise, let Δ be a derivation of (a,x) from G, given a rule-set R. Let $L \subseteq G$ be the set of the leaves of Δ. We say that Δ is *constrained with respect to rule-set* R iff $(a,\neg a) \notin deriv(L)$ where *deriv* is derivability using only rules in R. Equivalently, when R contains the rules AND, WO, iff $deriv(L,a)$ is consistent with a.

EXAMPLE 5. Put $G = \{(a,x), (a\wedge x,y)\}$ and let $R = R_3 = \{$SI, AND, WO, CT$\}$. Consider the following derivation Δ of $(a\wedge\neg x,y)$ from G.

$$\begin{array}{c} (a,x) \qquad\qquad (a\wedge x,y) \\ \text{----------} \quad \text{CT} \\ (a,y) \\ \mid \ \text{SI} \\ *(a\wedge\neg x,y) \end{array}$$

Δ is not constrained with respect to R because $(a\wedge\neg x,\neg(a\wedge\neg x)) \in deriv(L)$ where L is the set of leaves of Δ, as witnessed by the derivation:

$$\begin{array}{c} (a,x) \\ \mid \ \text{SI} \\ (a\wedge\neg x,x) \\ \mid \ \text{WO} \\ (a\wedge\neg x,\neg(a\wedge\neg x)) \end{array}$$

It is immediate from the definition that no derivation with root of the form $(a,\neg a)$ is constrained (with respect to the set of rules used in the derivation). If R also contains WO, then no derivation whose root (a,x) has an inconsistent fulfillment, is constrained. For if a is inconsistent with x then $x \vdash \neg a$ so that by an application of WO we have $(a,\neg a) \in deriv(L)$.

In the definition of a constrained derivation Δ of (a,x) from G given a rule-set R, we have considered only the set $L \subseteq G$ of leaves of Δ, but the whole set R of rules rather than those actually employed in Δ. It may be asked why we proceed in this asymmetric manner. We envisage the agent as working with a *fixed* stock R of derivation rules that it regards as acceptable. In general, a derivation will appeal to only some of the allowed stock of rules, but we regard the agent as remaining committed to all of them. On the other hand, the set G of pairs that the agent assumes as premises is regarded as *variable*. In general, a derivation will use only some of them as leaves, and they are the only ones to which the agent is committed. So when we test to see whether the body of the root is consistent with its own output under the rules, we do so with respect to the entire set R, but only the set $L \subseteq G$ of leaves of the derivation.

Evidently, this is a delicate point on which other perspectives are possible, with different consequences, for as L and R grow so does $deriv(L)$. We compare them in Appendix #4.

We say that (a,x) is *derivable with constraint* from G given rule-set R iff there is some derivation of (a,x) from G given R that is constrained with respect to R. As one would expect, for each of

our input/output operations, this is equivalent to full join constrained output, where the constraint is the same as the input.

OBSERVATION 9. Let *out* be any one of the operations out_i or out_i^+ (i = 1,..,4), and let R be the corresponding set of derivation rules. Then $x \in \cup(outfamily(G,a,a))$ iff there is a derivation of (a,x) from G (given rules from R) that is constrained (with respect to R).

Proof. Recall from Observation 1 that $x \in \cup(outfamily(G,a,a))$ iff $x \in out(H,a)$ for some $H \subseteq G$ such that a is consistent with $out(H,a)$. Since $out = deriv$ and the rules AND, WO are available, we need only show that the following are equivalent, where L_Δ is the set of all leaves of Δ.

(1) Δ is a derivation of (a,x) from some H with $H \subseteq G$, such that $\neg a \notin deriv(H,a)$,
(2) Δ is a derivation of (a,x) from G such that $\neg a \notin deriv(L_\Delta,a)$.

But (1) immediately implies $L_\Delta \subseteq H \subseteq G$ and so implies (2). Also (2) implies (1) taking H to be the set of all leaves of Δ other than those of the form (t,t) and, for out_i^+, of the form (a,a).

6.2. Truth-functional reductions of the constraint on derivations

We return to the question of truth-functional reductions of the input/output constraint, but now in terms of derivations and their leaves. Observation 8 gave a positive result for out_4 on the semantic level, which we can translate to the language of derivations.

OBSERVATION 10. Let Δ be any derivation of (a,x) from G, with leaves L. Then Δ is constrained with respect to R_4 iff a is consistent with $m(L)$.

Proof. By definition, Δ is constrained with respect to R_4 iff $\neg a \notin deriv_4(L,a) = out_4(L,a) = Cn(out_4(L,a))$, i.e. iff a is consistent with $out_4(L,a)$, i.e. (using Observation 8) iff a is consistent with $m(L)$.

Evidently, since $out_2^+(L,a) = out_4^+(L,a) = Cn(a \cup m(L))$ (section 1.1), Observation 10 also holds for R_2^+ and R_4^+.

For out_1 and out_3, the truth-functional reduction fails on the semantic level, as we saw in Examples 4.1 and 4.2. Nevertheless, on the level of derivations we have a positive result.

OBSERVATION 11. Let $R \in \{R_1,R_3\}$. Let Δ be any derivation of (a,x) from G with leaves L, given rules R. Then Δ is constrained with respect to R iff a is consistent (indifferently) with $m(L)$, $h(L)$, $f(L)$.

Proof. Given Observation 6, we need only show that if a is inconsistent with $f(L)$ then a is inconsistent with $deriv(L,a)$. For this, it suffices to show that $\{a\} \cup deriv(L,a) \vdash f(L)$.

For each node n:(b,y) in the derivation, write L_n for the set of all leaves in the subtree determined by n. We show by induction that $\{b\} \cup deriv(L,b) \vdash f(L_n)$.

Basis: Suppose n:(b,y) is a leaf of the tree. Then $y \in deriv(L,b)$, $f(L_n) = \{b \wedge y\}$, so clearly $\{b\} \cup deriv(L,b) \vdash f(L_n)$.

SI: Suppose n:(b,y) is derived by SI from p:(c,z). Then $b \vdash c$. By the induction hypothesis, $\{c\} \cup deriv(L,c) \vdash f(L_p)$, so $\{b\} \cup deriv(L,b) \vdash f(L_p) = f(L_n)$ since $L_n = L_p$.

AND: Suppose n:(b,y) is derived by AND from p:(b,z) and q:(b,w). By the induction hypothesis, $\{b\} \cup deriv(L,b) \vdash f(L_p), f(L_q)$ so $\{b\} \cup deriv(L,b) \vdash f(L_p) \cup f(L_q) = f(L_n)$ since $L_n = L_p \cup L_q$.

WO: Suppose n:(b,y) is derived by WO from p:(b,z). By the induction hypothesis, $\{b\} \cup deriv(L,b) \vdash f(L_p) = f(L_n)$ since $L_n = L_p$.

CT: Suppose n:(b,y) is derived by CT from p:(b,z) and q:($b \wedge z$,y). By the former, $z \in deriv(L,b)$ and, using CT, $deriv(L,b \wedge z) \subseteq deriv(L,b)$. By the induction hypothesis, $\{b\} \cup deriv(L,b) \vdash f(L_p)$ and $\{b \wedge z\} \cup deriv(L,b \wedge z) \vdash f(L_q)$. Putting these together we have $\{b\} \cup deriv(L,b) \vdash f(L_p) \cup f(L_q) = f(L_n)$ since $L_n = L_p \cup L_q$.

How does this Observation resist Examples 4.1 and 4.2? When considering constraints on a derivation, we look only at the set L of its leaves, and when $L \subset G$ then $m(L)$ may be weaker than $m(G)$. In the case of Example 4.1, where $G = \{(t,x), (x,y)\}$ and $A = \{\neg y\}$, we can construct a derivation of $(\neg y,x)$ with leaf-set $L = \{(t,x)\} \subset G$, as follows:

$$\begin{array}{l} (t,x) \\ \quad | \ \text{SI} \\ (\neg y,x) \end{array}$$

As noted in the presentation of Example 4.1, $\neg y$ is consistent with $out_1(G,\neg y) = deriv_1(G,\neg y) \supseteq deriv_1(L, \neg y)$. But while $\neg y$ is inconsistent with $m(G) = \{t \rightarrow x,\ x \rightarrow y\}$, it is consistent with $m(L) = \{t \rightarrow x\}$. In fact, there is no derivation of $(\neg y,x)$, using only rules from R_1, whose leaves cover all elements of G. Similar considerations apply to Example 4.2.

Observation 11 fails for R_2. Put G = $\{(a,\neg a),(b,\neg b)\}$ and consider the derivation:

$$\begin{array}{ccc} (a,\neg a) & & (b,\neg b) \\ \quad | \ \text{WO} & & \quad | \ \text{WO} \\ (a,\neg a \vee \neg b) & & (b,\neg a \vee \neg b) \\ \multicolumn{3}{c}{\text{----------}} & \text{OR} \\ & (a \vee b,\neg a \vee \neg b) & \end{array}$$

Then $a \vee b$ is inconsistent with $m(G)$. On the other hand, $a \vee b$ is consistent with $out_2(G,a \vee b)$, as witnessed by any complete set V containing a but not b. For then $a \vee b \in V$, and $G(V) = \{\neg a\}$ so $Cn(G(V)) = Cn(\neg a)$ which is consistent with $a \vee b$.

We end this section by noting that from Observation 11 we can get its counterpart for R_1^+ and R_3^+.

COROLLARY TO OBSERVATION 11. Let $R^+ \in \{R_1^+, R_3^+\}$. Let Δ be any derivation of (a,x) from G with leaves L, given rules R^+. Then Δ is constrained with respect to R^+ iff a is consistent (indifferently) with $m(L)$, $h(L)$, $f(L)$.

Proof. Let Δ be any derivation of (a,x) from G given rules R^+. Let L be the set of leaves of Δ. On the one hand, by Observation 6, if a is consistent with any one of $m(L)$, $h(L)$, $f(L)$ then it is consistent with $out_i^+(L,a)$, i.e. Δ is constrained wrt R^+. For the converse, note that since identity is a zero-premise rule, Δ is also a derivation, using only rules in R, from $G \cup J$ where $J \subseteq I$. If Δ is constrained wrt R^+ then it is constrained wrt to R and we may apply Observation 11 to conclude that a is consistent with each of $m(L)$, $h(L)$, $f(L)$.

7. APPLYING CONSTRAINTS MORE SEVERELY

7.1. General picture

If we are interested in derivations beyond their role as syntactic counterparts of explicitly defined operations, then further questions are suggested by the concepts introduced in Section 6. The definition of a constrained derivation (Section 6.1) requires that $(a,\neg a) \notin deriv(L)$, where a is the body of the root and L is the set of leaves of the derivation. But the root and the leaves are not the only nodes of a derivation. What happens if we apply the concept more severely? In particular:

- What if, instead of checking the body of the root node with respect to *leaves*, we check it *with respect to all nodes* in the derivation?

- What if, instead of checking the body *of the root node only*, we check *the body of every node* in the derivation?

The first question has an easy answer. It makes no difference whether we check the root with respect to leaves only, or all nodes in the derivation. This is irrespective of the subset of rules considered to be available.

OBSERVATION 12. Consider any set R of rules from SI, AND, WO, OR, CT. A derivation Δ with root (a,x) is constrained (with respect to R) iff $(a,\neg a) \notin deriv(H)$, where H is the set of all nodes of Δ (and derivability is defined in terms of R).

Proof. Since H is the set of all nodes of Δ, we have $L \subseteq H \subseteq deriv(L)$ where L is the set of leaves of Δ. Hence, since *deriv* (understood as taking sets of pairs of propositions to sets of pairs) is clearly a closure operation, we have $deriv(H) = deriv(L)$. In particular, $(a,\neg a) \notin deriv(H)$ iff $(a,\neg a) \notin deriv(L)$, i.e. iff Δ is constrained.

The second question is much more complex. The formulation above is rather loose. Let us say that a derivation Δ is *constrained at* node n:(b,y) iff $\neg b \notin deriv(L,b)$, where L is the set of leaves of the entire derivation Δ and *deriv* is defined by the same set of rules as used when checking the

root node. Our question is: given a derivation of (a,x) from G that is constrained at the root, is there always some derivation of (a,x) from G that is constrained at every node?

In general, the answer depends on the rules allowed, for one rule may allow us to bypass steps that fall foul of a constraint which another rule cannot avoid. Specifically, for the rule sets R_2 and R_4, it can make a difference to derivations whether we check the root only, or all nodes in the derivation. But for the rule sets R_1 and R_3 (which lack OR), it makes no difference. Moreover, for the rule set R_2 (which lacks CT), if the given derivation satisfies the constraint at *both* the root *and* all leaves, then *some* derivation of the same root from the same leaves satisfies the constraint at every step. We now prove these results.

7.2. Results on severe application of the constraint on derivations

We begin with the negative result, which reveals the limits of the positive ones that follow.

OBSERVATION 13. For rule set $R \in \{R_2, R_4\}$, there is a derivation that is constrained with respect to R, but such that no derivation given R of the same root from the same generators is constrained at every node with respect to R.

Proof. We re-use Example 2.1 of Appendix #1. Put $G = \{(a,\bot), (\neg a,x)\}$ and consider the following derivation.

```
*(a,⊥)                    (¬a,x)
   |  WO                    |
 *(a,x)                     |
 --------------------------   OR
        (a∨¬a,x)
```

This derivation is constrained with respect to R_2, since $\neg(a\vee b) \notin deriv_2(G,a\vee\neg a) = deriv_4(G,a\vee\neg a) = Cn(x)$. But it fails the constraint at the starred nodes. Moreover, there is no other derivation of the same root from the same (or fewer) leaves that satisfies the constraint at all nodes. For if the constraint is to be satisfied at all nodes, the leaf $(a,\bot)$ cannot be used; but the root is not in even the unrestricted output of the other leaf $(\neg a,x)$ taken alone, as is easily checked from its definition (Section 1.1).

We now pass to the positive results. We begin by noting some preservation properties for the various rules.

LEMMA 14. Let {SI, AND, WO} $\subseteq R \subseteq$ {SI, AND, WO, OR, CT}, i.e. let R be one of R_i (i = 1,2,3,4). Then:

(a) Satisfaction of the constraint is preserved backwards by each of the rules SI, AND, WO, CT. That is, in any derivation, if the conclusion of the rule satisfies the constraint (with respect to R), then so do its premises.

(b) Satisfaction of the constraint is preserved forwards by each of the rules AND, WO, CT, OR. That is, in any derivation, if the premises of the rule satisfy the constraint (with respect to R), then so does its conclusion.

Proof.
SI: Suppose (b,x) is obtained from (a,x) where $b \vdash a$. Suppose the premise fails the constraint. Then $\neg a \in deriv(L,a) \subseteq deriv(L,b)$. But since $b \vdash a$ we have $\neg a \vdash \neg b$ so since WO is in R, $\neg b \in out(L,b)$ and the conclusion fails the check.

AND: Suppose $(a,x\wedge y)$ is obtained from (a,x) and (a,y). The conclusion has the same body as each of the two premises, and so we have preservation in both directions.

WO: Suppose (a,y) is obtained from (a,x) where $x \vdash y$. Again, the conclusion has the same body as the premise, and so we have preservation in both directions.

CT: Suppose (a,y) is obtained from (a,x) and $(a\wedge x,y)$. Preservation forwards is immediate, since the conclusion has the same body as one of the two premises. For preservation backwards, we have two cases to consider. Suppose first that (a,x) fails the constraint. Then $\neg a \in deriv(L,a)$ and the conclusion fails the constraint. Suppose next that $(a\wedge x,y)$ fails the constraint. Then $\neg(a\wedge x) \in deriv(L,a\wedge x)$. But also $x \in deriv(L,a)$, so by CT, $\neg(a\wedge x) \in deriv(L,a)$. So by AND, WO which are in R, $\neg a \in deriv(L,a)$ and the conclusion fails the constraint.

OR: Suppose $(a\vee b,x)$ is obtained from (a,x) and (b,x). Suppose that the conclusion fails the constraint, i.e. $\neg(a\vee b) \in deriv(L,a\vee b)$. Then since SI is in R, $\neg(a\vee b) \in deriv(L,a)$ say, so since WO is in R, $\neg a \in deriv(L,a)$ and the premise fails the constraint.

We note in passing that that Lemma 14(a) continues to apply if the value of L is not fixed, but is taken to be the set of leaves in the derivation up to the node being considered. However, parts of Lemma 14(b) would then fail. As shown by Examples 7.1-7.3 in Appendix #5, satisfaction would no longer be preserved forwards for AND, CT, OR.

From Lemma 14 we have immediately the following for derivations without OR:

OBSERVATION 15. For $R \in \{R_1,R_3\}$, if the constraint is satisfied at the root then it is satisfied at every node. More explicitly: let $R \in \{R_1,R_3\}$, and let Δ be any derivation using only rules from R that satisfies the constraint (with respect to R) applied to the root. Then Δ satisfies the same constraint (with respect to R) at every step.

The example used in the proof of Observation 13 already shows that Observation 15 can fail when OR is present. But we can also obtain a weaker result for derivations with OR, so long as they do not contain CT.

OBSERVATION 16. For derivations without CT, satisfaction of the constraint at both root and leaves suffices to ensure its satisfaction everywhere in some derivation of the same root from the same leaves. More explicitly: let $R \in \{R_1,R_2\}$, and let Δ be any derivation given R, that satisfies the constraint (with respect to R) applied to the root and also applied to each leaf. Then there is a

derivation Δ' given R with the same root and leaves that satisfies the constraint at every node.

Proof. By Observation 19(b) of (Makinson and van der Torre, 2000) any unrestricted derivation using only the given rules may be rewritten as one in which they are applied in the order WO, OR, SI, AND. Here it is understood that some of the rules may be skipped, and some may be applied several times, but never applied contrary to the indicated order. Put Δ' to be a derivation ordered in this way. As the leaves and the root have not changed, the constraint continues to be satisfied for Δ'. It remains to show that the constraint is satisfied at every node of Δ'. For this we need only apply Lemma 14(b) forwards from the leaves through WO, OR, and Lemma 14(a) backwards from the root through AND, SI, thus covering all nodes in Δ'.

Observation 16 cannot be extended to derivations admitting both OR and CT. More explicitly:

OBSERVATION 17. For the rule-set R_4: there is a derivation that satisfies the constraint applied to the root and also applied to each leaf, but such that neither it nor any other derivation given R_4 of the same root from the same generators (or a subset of them) satisfies the constraint at every node.

Proof. See Appendix #6.

8. SUMMARY AND PROSPECTS

8.1. Summary

In this paper, we have studied what happens when input/output operations are constrained to avoid excess output. Our strategy for eliminating excess output is to cut back the set of generators to just below the threshold of yielding excess. To do that, we adapt a technique that is well known in the more specific areas of belief change and nonmonotonic inference – look at the maximal subsets of the generator set whose output is not excessive.

Outfamily(G,A,C) is defined as the family of all sets *out*(H,A) where H is a maximal subset of G such that *out*(H,A) is consistent with C. The case $C = \varnothing$ corresponds to the requirement that output is consistent; the case $C = A$ to consistency of output with input. When the underlying unrestricted input/output operation is reusable basic throughput (out_4^+), this gives exactly the default extensions of Poole and the maxichoice revisions of Alchourrón and Makinson (Observation 4). When it is reusable simple-minded throughput (out_3^+), the result is very closely related to the normal default extensions of Reiter (Observation 5).

Our main focus in the paper is on the case where output is required to be consistent with input. We call this *the input/output constraint*. In the context of reusable basic output out_4, the definition of *maxfamily*(G,A,A) can be given a 'truth-functional reduction' in terms of the materialisation of the generating set (Observation 8).

Constraints may also be approached in terms of derivations. Here, the natural constraint is to require the body a of the root (a,x) of the derivation to be consistent with its own unrestricted

output under the leaves of the derivation. This amounts to requiring that $x \in \cup(outfamily(L,a,a))$ (Observation 9). In the context of derivations, we can give more truth-functional reductions of the constraint than we could on the semantic level (Observations 10 and 11).

Finally, we investigated the consequences of applying the consistency constraint more severely than at the root only and with respect to the leaves only. On the one hand, if the constraint is applied at the root only, but with respect to all nodes, this makes no difference (Observation 12). On the other hand, if we apply the constraint at all nodes with respect to the leaves, it can make a difference (Observation 13).

Nevertheless, when the choice of derivation rules is limited in certain ways, application of the constraint at all nodes of a derivation no longer increases its severity. This is so for derivations without OR (Observation 15), because all rules from our palette other than OR preserve backwards satisfaction of the constraint (Lemma 14a). For derivations without CT, the situation is subtler. The rules other than CT may be partitioned into those that preserve forward satisfaction of the constraint and those that preserve backward satisfaction (Lemma 14), and derivations without CT can always be rewritten with the former rules applied first. Thus checking the constraint at both root and leaves suffices to guarantee its satisfaction at every step of the rewritten derivation (Observation 16).

8.2. Prospects

A technical problem that has been left open is the order structure of outfamilies for out_i where i = 2,3,4, and to a lesser extent for out_3^+, specifically the role of their maximal elements (see end of Section 4 and Appendix #2).

Some general lines of investigation deserve further exploration. These include the properties of full and partial meets and unions of outsets (cf. the observations on monotony and antitony in Appendix #1), and constraints with respect to values of C other than A and $\varnothing$. One might also consider possible refinements in the definition of an outfamily. For example, adapting an idea already studied in the logic of belief revision (Makinson 1997), one could designate a certain part of the unrestricted output as protected from attrition in the construction of outfamilies.

On the level of derivations, it could be of interest to investigate ways of applying the constraint at every step, but with respect to a variable set L of leaves (cf. the remarks after Lemma 14, and the examples in Appendix #5). One might also try to ascertain whether our choice of rules SI, WO, AND, OR, CT provides the best possible palette for the analysis of derivations, or whether there are other sets of rules, collectively equivalent but separately not so, that give more insight into the powers of consistency constraints.

One could also go back to unconstrained systems, and ask whether there are others, different from those studied here, that merit examination. For application to defeasible obligations, it may be of interest to study systems that do not validate SI. For application to permissive norms (even on the indefeasible level) one could consider systems failing both SI and AND.

Finally, we recall that the paper focuses on the creation and study of an abstract structure. It

remains to consider its application in practical contexts. Some well-known examples of contrary-to-duty conditional norms are analysed in Section 3 (Examples 3.1-3.3), but there are many more – see for instance the tables in (Makinson 1999) and (van der Torre and Tan 1999). Application elsewhere, such as the logics of action and belief, remains open.

APPENDICES

#1. Monotony/antitony Properties for the Full Join and Meet Operations

It is immediate from Observation 1 that the full join operation is monotonic in argument G and antitonic in C. For argument A, the situation is less stable, depending on the choice of the underlying unrestricted input/output operation.

On the one hand, the full join operation based on out_1 is monotonic in A. For suppose $x \in \cup(outfamily(G,A,C))$. By Observation 1, there is a $H \subseteq G$ such that $out_1(H,A) = Cn(H(Cn(A)))$ contains x and is consistent with C. We may assume without loss of generality that all bodies of elements of H are in $Cn(A)$, since cutting them out makes no difference to the value of $H(Cn(A))$. Thus when $A \subseteq B$ we have $H(Cn(A)) = H(Cn(B))$, so that $out_1(H,B) = Cn(H(Cn(B))) = Cn(H(Cn(A)))$ is consistent with C and contains x, so we may apply Observation 1 again and conclude. A more complex version of this argument shows the same for underlying out_3.

On the other hand, when the underlying unconstrained operation is out_2 or out_4, then the full join operation fails monotony in A, as shown in the following example.

EXAMPLE 2.1. Put $G = \{(a,\bot), (\neg a,x)\}$ and let $out \in \{out_2, out_4\}$. We show $x \in \cup(outfamily(G,Cn(t),\emptyset))$ while $x \notin \cup(outfamily(G,Cn(a),\emptyset))$ although $Cn(t) \subseteq Cn(a)$. On the one hand, $out(G,t) = Cn(x)$ which is consistent, so $maxfamily(G,t,\emptyset) = \{G\}$ and $outfamily(G,t,\emptyset) = \{Cn(x)\}$ and thus $\cup(outfamily(G,t,\emptyset)) = Cn(x)$. On the other hand $out(G,a) = Cn(\bot)$, which is inconsistent so $maxfamily(G,a,\emptyset) = \{(\neg a,x)\}$, so $outfamily(G,a,\emptyset) = \{Cn(\emptyset)\}$ and thus $x \notin \cup(outfamily(G,a,\emptyset)) = Cn(\emptyset)$.

Again, monotony in A fails when the underlying unconstrained operation is any of out_n^+ authorising input to reappear as output.

EXAMPLE 2.2. Put $G = \{(t,x)\}$. For $out = out_n^+$ we have $out(G,t) = Cn(x)$ which is consistent so $\cup(outfamily(G,a,\emptyset)) = Cn(x)$, while $out(G,\neg x) = Cn(\bot)$, so $maxfamily(G,\neg x,\emptyset) = \{\emptyset\}$ and $outfamily(G,\neg x,\emptyset)) = \{Cn(\emptyset)\}$ and thus $\cup(outfamily(G,\neg x,\emptyset)) = Cn(\emptyset)$.

The full meet operation is even less well behaved. It fails all three properties: monotony in G, antitony in C, and monotony in A. The following examples are calculated indifferently for our four input/output operations without throughput.

EXAMPLE 2.3. To illustrate non-monotony in argument G for the full meet operation, put $G = \{(a,x)\}$. Then $x \in Cn(x) = \cap(outfamily(G,a,\emptyset))$. But for $H = G\cup\{(a,\neg x)\}$, the set $out(H,a)$ is

inconsistent and $maxfamily(H,a,\emptyset) = \{\{(a,x)\},\{(a,\neg x)\}\}$, so that $\cap(outfamily(H,a,\emptyset)) = Cn(x)\cap Cn(\neg x) = Cn(\emptyset)$, which does not contain x.

EXAMPLE 2.4. To illustrate non-monotony in argument A, put $G = \{(a,x), (b,\neg x)\}$. Then $x \in Cn(x) = \cap(outfamily(G,a,\emptyset))$. But $\cap(outfamily(G,\{a,b\},\emptyset)) = Cn(x)\cap Cn(\neg x) = Cn(\emptyset)$, which does not contain x.

EXAMPLE 2.5. To illustrate failure of antitony in argument C, put $G = \{(a,x), (a,y)\}$, $C = \{\neg x\vee\neg y\}$, $D = C\cup\{\neg x\}$. Then $C \subseteq D$, and $y \in Cn(y) = \cap(outfamily(G,a,D))$, but $y \notin Cn(x)\cap Cn(y) = \cap(outfamily(G,a,C))$.

#2. Elements of outfamily versus maximal outputs

It is important to distinguish between an outfamily, i.e. the set of outputs determined by maximal subsets of the generators, and the set of maximal outputs. The latter is a subset of the former, but for certain of the input/output operations, the two sets are not identical.

OBSERVATION 2. Let X be a maximal value of $out(H,A)$, for H ranging over subsets of G such that $out(H,A)$ is consistent with C. Then $X \in outfamily(G,A)$.

Sketch of Proof. Straightforward, using the compactness of each of our input/output operations, established in (Makinson and van der Torre, 2000).

On the other hand, the converse fails for certain of the input/output operations. For $out \in \{out_3, out_3^+, out_4\}$ the Möbius strip provides a counterexample, as shown in Example 3.3 and in the discussion of normal Reiter default systems (Section 4).

For out_2, the same phenomenon can be illustrated by a different example. Put $G = \{(a,\bot), (\neg a,\bot), (\neg a,x)\}$. Then, calculating for $out = out_2$ with t as input and $\emptyset$ as constraint, $out(G,t) = Cn(\bot)$, which is inconsistent, and $maxfamily(G,t,\emptyset) = \{\{(a,\bot),(\neg a,x)\}, \{(\neg a,\bot),(\neg a,x)\}\}$ so that $outfamily(G,t,\emptyset) = \{Cn(x), Cn(\emptyset)\}$, and evidently $Cn(x) \supset Cn(\emptyset)$.

But for out_1, out_1^+, and out_4^+ (alias out_2^+) we get the opposite. In those cases, every element of outfamily is maximal, as we now show.

OBSERVATION 3. Let $out = out_i$ for $i \in \{1,1^+,2^+,4^+\}$. Then no element of $outfamily(G,A,C)$ properly includes any other.

Proof. Let X, Y be elements of $outfamily(G,A,C)$ and suppose $X \subset Y$; we derive a contradiction. Since $X, Y \in outfamily(G,A,C)$ we have:

> $X = out(G_1,A)$, where $G_1 \subseteq G$, X is consistent with C; and for all G_1' with $G_1 \subset G_1' \subseteq G$, $out(G_1',A)$ is inconsistent with C

$Y = out(G_2,A)$, where $G_2 \subseteq G$, Y is consistent with C; and for all G_2' with $G_2 \subset G_2' \subseteq G$, $out(G_2',A)$ is inconsistent with C.

Since $X \subset Y$, we have $X \subseteq Y$, i.e. $out(G_1,A) \subseteq out(G_2,A)$. Also since $X \subset Y$, we have not: $Y \subseteq X$, so not: $G_2 \subseteq G_1$, so $G_1 \subset G_1 \cup G_2$. Hence by maximality of G_1, $out(G_1 \cup G_2,A)$ is inconsistent with C. Thus to obtain a contradiction, it suffices to show that $out(G_1 \cup G_2,A) \subseteq out(G_2,A)$ which by hypothesis is consistent with C.

For out_1, we branch the argument as follows. Since $out(G_1,A) \subseteq out(G_2,A)$ we have by the definition of out_1 in Section 1.1 that $Cn(G_1(Cn(A))) \subseteq Cn(G_2(Cn(A)))$ so $G_1(Cn(A)) \subseteq Cn(G_2(Cn(A)))$. Clearly also $G_2(Cn(A)) \subseteq Cn(G_2(Cn(A)))$. Thus $G_1 \cup G_2(Cn(A)) \subseteq Cn(G_2(Cn(A)))$ and so finally $Cn(G_1 \cup G_2(Cn(A))) \subseteq Cn(G_2(Cn(A)))$ i.e. $out(G_1 \cup G_2,A) \subseteq out(G_2,A)$ as desired. For out_1^+, simply replace G_1,G_2 by $G_1 \cup I, G_2 \cup I$ in this argument.

For out_4^+ alias out_2^+, we branch as follows. Since $out(G_1,A) \subseteq out(G_2,A)$ we have by the characterization in Section 1.1 of $out_4^+(G,A)$ as $Cn(A \cup m(G))$, that $Cn(A \cup m(G_1)) \subseteq Cn(A \cup m(G_2))$ so $out(G_1 \cup G_2,A) = Cn(A \cup m(G_1) \cup m(G_2)) \subseteq Cn(A \cup m(G_2)) = out(G_2,A)$ as desired.

#3. Normal Reiter Default Systems: Proof of Observation 5.

We prove Observation 5 from first principles, not having seen any result in the literature from which it would follow directly. The argument is not difficult, but rather complex when set out rigorously.

Proof. We need to prove assertions (a) and (b) of the Observation. For (a), let E be any extension of the normal Reiter default system (G,A). To show that $E \in outfamily(G,A)$ it suffices to show that $E = out_3^+(H,A)$ for some maximal $H \subseteq G$ such that $out_3^+(H,A)$ is consistent.

Put $H = \{(a,x) \in G$: either $a \notin E$ or x is consistent with $E\}$. Then $H \subseteq G$, and it is easy to show using the quasi-inductive characterization of extensions in Theorem 2.1 of (Reiter 1980) that E is also an extension of the normal Reiter default system (H,A). Since A is consistent, we know from Corollary 2.2 of (Reiter 1980) that E is consistent. It remains to show that (1) $E = out_3^+(H,A)$, (2) $out_3^+(H',A)$ is inconsistent for all H' with $H \subset H' \subseteq G$.

For (1), by Reiter's fixed-point definition of an extension, $E = \cap\{B$: $A \subseteq B = Cn(B)$, such that $x \in B$ whenever $(a,x) \in H$ and $a \in B$ and x is consistent with $E\}$. On the one hand, since out_3^+ is reusable throughput, $out_3^+(H,A)$ is such a B (without even appealing to the consistency condition), so $E \subseteq out_3^+(H,A)$. On the other hand, E is also such a B, i.e. $A \subseteq E = Cn(E)$, and $x \in E$ whenever $(a,x) \in H$ and $a \in E$ and x is consistent with E. But by the definition of H, $(a,x) \in H$ and $a \in E$ together imply that x is consistent with E. Thus $A \subseteq E = Cn(E)$, and $x \in E$ whenever $(a,x) \in H$ and $a \in E$, i.e. $A \subseteq E = Cn(E) \supseteq H(E)$, so $E \supseteq \cap\{B$: $A \subseteq B = Cn(B) \supseteq H(B)\} = out_3^+(H,A)$ by a characterization of the latter (Section 1.1).

To show (2), suppose $H \subset H' \subseteq G$ with $(a,x) \in H'-H$, so that by the definition of H, $a \in E$ and x is inconsistent with E. By (1), $E = out_3^+(H,A) \subseteq out_3^+(H',A)$, so x is inconsistent with $out_3^+(H',A)$.

But since $(a,x) \in H'$ and $a \in E \subseteq out_3^+(H',A)$ we have $x \in out_3^+(H',A)$, since out_3^+ satisfies reusability. Putting these together, $out_3^+(H',A)$ is itself inconsistent as desired.

To show assertion (b), let $X \in outfamily(G,A)$. Then $X = out_3^+(H,A)$ for some maximal $H \subseteq G$ such that $out_3^+(H,A)$ is consistent. It will suffice to show that X is an extension of the normal Reiter default system (H,A), for as shown in the 'semi-monotonicity' Theorem 3.2 of (Reiter 1980), when H,G are sets of normal default rules with $H \subseteq G$, then every extension of system (H,A) is included in some extension of (G,A).

By the hypothesis, X is consistent. Consider the sequence $E_0, E_1, \ldots$ where $E_0 = A$ and $E_{i+1} = \mathrm{Cn}(E_i) \cup \{x: (a,x) \in H \text{ and } a \in E_i \text{ and } x \text{ is consistent with } X\}$. By Theorem 2.1 of (Reiter 1980), to show that X is an extension of (H,A), it suffices to show that $\cup\{E_i : 0 \leq i < \omega\}$, written briefly $\cup E_i$, equals X.

By induction, we have each $E_i \subseteq out_3^+(H,A) = X$ so that $\cup E_i \subseteq X$. For the converse, we have by a characterization of out_3^+ (Section 1.1) that $out_3^+(H,A) = \cap\{B: A \subseteq B = Cn(B) \supseteq H(B)\}$ so it suffices to show that $\cup E_i$ is such a B, i.e. that $A \subseteq \cup E_i = Cn(\cup E_i) \supseteq H(\cup E_i)$. The first inclusion and the equality are trivial. To show the second inclusion, suppose $x \in H(\cup E_i)$ so that $(a,x) \in H$ for some $a \in \cup E_i$, so $a \in E_i$ for some i. It will suffice to show $x \in E_{i+1}$. By the definition of E_{i+1} it suffices to show that x is consistent with X. But since $a \in E_i \subseteq out_3^+(H,A) = X$ and $(a,x) \in H$ we have $x \in X$ by reusability, so since X is consistent, x is consistent with X as desired and we are done.

#4. Remarks on the definition of a constrained derivation

Let Δ be a derivation of (a,x) from G given a rule-set R, and let $L \subseteq G$ be the set of leaves of Δ. In Section 6.1 we defined Δ to be *constrained with respect to rule-set R* iff $(a,\neg a) \notin deriv(L)$ where *deriv* is derivability using only rules in R.

To illustrate the effect of using $deriv(L)$ rather than $deriv(G)$ in this definition, put $G = \{(a,x), (a,y)\}$ and consider the one-step derivation:

$$\begin{array}{c} (a,x) \\ \Big| \ \text{SI} \\ (a\wedge\neg y,x). \end{array}$$

This derivation is constrained under our definition, since $(a\wedge\neg y, \neg(a\wedge\neg y) \notin deriv(L)$. But $(a\wedge\neg y, \neg(a\wedge\neg y) \in deriv(G)$, by applying SI and WO to the other element of G.

To illustrate the effect of using the whole of R rather than the subset of its rules actually applied in Δ, consider again Example 5 in Section 6.1. On the one hand, it is not difficult to show that $(a\wedge\neg x,\neg(a\wedge\neg x)) \notin deriv(L)$ when *deriv* is determined by the set {SI, CT} of rules actually applied in Δ. On the other hand, as noted in Example 5, Δ is not constrained with respect R_3 = {SI, AND, WO, CT}, since $(a\wedge\neg x,\neg(a\wedge\neg x)) \in deriv(L)$ where *deriv* is determined by that rule-set. A more interesting illustration of the same point is the following.

EXAMPLE 6. Put $G = \{(a, x\wedge(a\rightarrow y)), (\neg a, x\wedge(\neg a\rightarrow y))\}$ and consider the derivation:

$$
\begin{array}{ccc}
(a, x\wedge(a\rightarrow y)) & & (\neg a, x\wedge(\neg a\rightarrow y)) \\
| \quad \text{WO} & & | \quad \text{WO} \\
(a,x) & & (\neg a,x) \\
\multicolumn{3}{c}{\text{----------}} \quad \text{OR} \\
\multicolumn{3}{c}{(t,x)} \\
\multicolumn{3}{c}{| \quad \text{SI}} \\
\multicolumn{3}{c}{(\neg y,x)}
\end{array}
$$

This derivation is constrained with respect to the set {SI, WO, OR} of rules applied within it. On the other hand, it is not constrained with respect to the larger set R_4 = {SI, WO, AND, OR, CT}. Indeed, it is not difficult to show that there is no derivation, given R_4, of the same root from the same leaves (or a subset of them), that is constrained with respect to R_4. We omit the verifications.

#5. Varying the set L and its effect on Lemma 14

Suppose that when constraining a node n:(a,x) of a derivation we do so with respect to the set L_n of leaves in its subtree, rather than the set L of all leaves of the tree (see remark after Lemma 14). Then parts of Lemma 14(b) fail: satisfaction of the constraint is no longer preserved forwards by the rules AND, CT, OR. We give an example for each. In each example, the derivation fails the constraint at the conclusion of the rule in question, marked by an asterisk. It also fails the constraint at the premises of that rule, if L is held constant. But it would satisfy the constraint at each premise of the rule if L were allowed to diminish. For instance in Example 7.1, node n:(a,x), a premise of the AND rule, is a leaf and so $L_n = \{(a,x)\}$, and $(a,\neg a) \notin deriv(L_n)$.

EXAMPLE 7.1 (for AND). Let $R \in \{R_1,..,R_4\}$ and consider the following derivation.

$$
\begin{array}{ccc}
(a, x) & & (a, \neg x) \\
| & & | \\
\multicolumn{3}{c}{\text{----------}} \quad \text{AND} \\
\multicolumn{3}{c}{*(a,\perp)}
\end{array}
$$

EXAMPLE 7.2 (for CT). Let $R \in \{R_3,R_4\}$ and consider the following derivation.

$$
\begin{array}{ccc}
(a, x\wedge\neg y) & & \\
| \quad \text{WO} & & \\
(a,x) & & (a\wedge x,y) \\
\multicolumn{3}{c}{\text{----------}} \quad \text{CT} \\
\multicolumn{3}{c}{*(a,y)}
\end{array}
$$

EXAMPLE 7.3 (for OR). Let $R \in \{R_2,R_4\}$ and consider the following derivation.

$$
\begin{array}{ccccc}
(a, x) & & (a,\neg x) & & \\
| & \text{WO} & | & \text{WO} & \\
(a, t) & & (a, t) & & \\
\multicolumn{3}{c}{\text{----------}} & \text{OR} & \\
\multicolumn{3}{c}{*(a, t)} & &
\end{array}
$$

#6. Derivations using both OR and CT: proof of Observation 17

Proof. Consider the following derivation of $((a\wedge\neg x)\vee b,\ x\rightarrow\neg(a\wedge b))$ from the generator set $G = \{(a,x), (b, x\rightarrow\neg(a\wedge b))\}$.

$$
\begin{array}{ccccc}
(a,x) & & (b, x\rightarrow\neg(a\wedge b)) & & (b, x\rightarrow\neg(a\wedge b)) \\
| & \text{SI} & | & \text{SI} & | \\
* (a\wedge(x\rightarrow b)), x) & & *(a\wedge b\wedge x), x\rightarrow\neg(a\wedge b)) & & | \\
\multicolumn{4}{c}{\text{----------}} & \text{CT} \quad | \\
\multicolumn{4}{c}{*(a\wedge(x\rightarrow b)), x\rightarrow\neg(a\wedge b))} & & | \\
\multicolumn{4}{c}{\text{----------}} & \text{OR} \\
\multicolumn{4}{c}{((a\wedge\neg x)\vee b, x\rightarrow\neg(a\wedge b))} & &
\end{array}
$$

On the one hand, the derivation satisfies the constraint at the root and at the leaves. By Observation 10, to show this it suffices to check that the body of the root, and the body of each leaf, is consistent with $m(L) = m(G)$. On the other hand, the derivation fails the constraint at each of the starred nodes. By Observation 10 again, it suffices to check that the body of each starred node is inconsistent with $m(L)$. We now show that there is *no derivation* with the same root and the same (or fewer) leaves that satisfies the constraint at all nodes. This is the challenging part of the proof.

Call a node of a derivation *small* if it is in the unrestricted output of a single leaf; otherwise *big*. Quite generally, if a node is small, then it is of the form (c^+,z^-) for some leaf (c,z) where $c^+ \vdash c$ and $z \vdash z^-$, as can be checked by simple induction on the derivation.

Consider any derivation, given rule-set R_4, of the root node from G. Its leaves will all be from G, or else of the form (t,t) where t is a tautology. The root node is big, since it is not of the above form. Hence there is a first big node in the derivation. This need not be unique, but we choose one and call it m. By construction, m is not one of the leaves, so it must be obtained using one of the rules SI, WO, AND, CT, OR. Since m is a first big node, it is not obtained by the single-premise rules SI or WO, so it must be obtained by one of AND, CT, OR from two small premises. If either of these two premises is of the form (d,t) then m will be equivalent to the other premise, or else itself of the form (e,t) as is easily checked by cases; and so will itself be small, contrary to supposition. So m is obtained by one of AND, CT, OR from two small premises, one of which is of the form (a^+,x^-) and the other of the form $(b^+, (x\rightarrow\neg(a\wedge b))^-)$.

Application of OR to two such premises gives a conclusion of the form (d,t) which is small, leaving only the rules AND, CT to consider.

Application of AND to two such premises is possible only if a^+ is equivalent to b^+, in which case $a^+ \vdash a \wedge b$. But it is easily checked that $\neg(a \wedge b) \in out_4(G, a \wedge b)$, so by SI and WO, $\neg\, a^+ \in out_4(G, a^+)$, so node *m* fails the constraint (as also its two premises).

For CT there are two cases to consider, as it is an asymmetric rule. In the first case, b^+ is equivalent to $a^+ \wedge x^-$, so that $b^+ \vdash a \wedge b$ so by the same argument as for AND, the premise with body b^+ fails the constraint. In the second case, a^+ is equivalent to $b^+ \wedge (x \rightarrow \neg(a \wedge b))^-$ so $a^+ \vdash a \wedge b$ and by the same argument the premise with body a^+ fails the constraint.

REFERENCES

Alchourrón, Carlos, Peter Gärdenfors and David Makinson, 1985. On the logic of theory change: partial meet contraction and revision functions, *The Journal of Symbolic Logic* 50: 510-530.

Alchourrón, Carlos and David Makinson, 1982. On the logic of theory change: contraction functions and their associated revision functions, *Theoria* 48: 14-37.

Hansson, Bengt, 1969. An analysis of some deontic logics, *Nous* 3: 373-398. Reprinted in R. Hilpinen, R. ed *Deontic Logic: Introductory and Systematic Readings*, Dordrecht: Reidel, 1971 and 1981, pp. 121-147.

Hansson, Sven Ove and David Makinson, 1997. Applying normative rules with restraint, in M.L. Dalla Chiara et al eds *Logic and Scientific Methods*, Dordrecht: Kluwer, pp. 313-332.

Makinson, David, 1994. General patterns in nonmonotonic reasoning. In Dov Gabbay et al eds *Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 3*, Oxford University Press, pp. 35-110.

Makinson, David, 1997. Screened revision, *Theoria* 63: 14-23.

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

Makinson, David and Leendert van der Torre, 2000. Input/output logics, *J. Philosophical Logic* 29: 383-408.

Poole, David, 1988. A logical framework for default reasoning, *Artificial Intelligence* 36: 27-47.

Reiter, Ray. 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, in *Logics in Artificial Intelligence*, Proceedings of the Sixth European Workshop on Logics in AI (JELIA'98). Berlin: Springer, LNCS 1489, pp 92-106.

van der Torre, Leendert W.N. and Yao-Hua Tan, 1999. Contrary-to-duty reasoning with preference-based dyadic obligations. *Annals of Mathematics of Artificial Intelligence* 27: 49-78.

ACKNOWLEDGEMENTS

Thanks for helpful comments to Salem Benferhat, the referees for DEON 2000 where an early version of this paper was presented, and the referee for this journal. Research for the 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: dcmakinson@aol.com

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

Last revised 11.01.01
Typos corrected 20.01.01
Word count: 12,616