Introducing Exclusion Logic as a Deontic Logic

Richard Evans*

Maxis

Abstract. This paper introduces Exclusion Logic - a simple modal logic without negation or disjunction. We show that this logic has an efficient decision procedure. We describe how Exclusion Logic can be used as a deontic logic. We compare this deontic logic with Standard Deontic Logic and with more syntactically restricted logics.

Key words: deontic, material incompatibility, exclusion, hierarchical finite-state machine

1 Outline

We start with a simple pre-propositional system called Exclusion Logic (hereafter **EL**). Once we have outlined the semantics and decision-procedure for EL, we use it to define a specifically deontic exclusion logic, **DEL** as a syntactic extension of EL.

EL was designed to capture the expressive resources of Hierarchical Finite-State Machines in a declarative language. It is a simple language for constructing trees of data. It does not contain operators for negation or disjunction, and uses only a restricted version of implication. Instead, it has an operator for expressing the idea that one data-value *excludes* other values.

1.1 Motivation

Typically, logicians treat relations of material-incompatibility between propositions as defined in terms of non-logical rules involving negation. For example: if we want to say that Red and Green are one-place predicates which are materially incompatible, we have to add extra non-logical axioms:

$$Red(x) \rightarrow \neg\, Green(x)$$
$$Green(x) \rightarrow \neg\, Red(x)$$

But there is an alternative tradition, initiated by Hegel and rearticulated by Sellars and Brandom, in which *material incompatibility is conceptually prior to negation.* According to this alternative tradition, there can be linguistic practices

* Thanks to Martin Berger and Jeff Orkin for detailed feedback. I am also grateful to the Berkeley Social Ontology Group, in particular John Searle, Asa Andersen, Maya Kronfeld and Raffaela Giovagnoli for feedback on a previous draft.

in which users treat predicates such as $Green(x)$ and $Red(x)$ as incompatible, but which contain no operator for general negation.

EL is a logic which respects this alternative direction of explanation. It uses the exclusion operator "." to express materially-incompatible propositions without the need for an explicit negation operator and non-logical axioms.

1.2 Syntax

Definition 1. *Given a set S of symbols, the expressions in EL are defined as all terms of type G in:*

$$X ::= S \mid S{:}X \mid S.X$$
$$E ::= X \mid E \wedge E$$
$$G ::= E \mid E \rightarrow E$$

The language is carefully stratified: terms are assembled into conjunctions, which are then combined into implications. In this way, it resembles disjunctive normal form. Note that $\rightarrow$ is not recursively embeddable: although $P \rightarrow Q$ is well-formed, $P \rightarrow Q \rightarrow R$ is not.

Remark 1. The only unusual feature of this simple language is the use of two binary modal operators, '.' and ':', to build up trees of information. Asserting that $A{:}B$ is claiming both that A, and that *one* of the ways in which A is B. Saying that $A.B$, by contrast, is to say that B is the *only* way in which A is the case.

Terms	Explanation
WhoseMove.Black	It is Black to move
Agents:Jack:Age.37	One of the agents is Jack, and his age is 37
Agents.Jack:Age.38	Jack is the only agent, and his age is 38
Wolf:Class.Mammalia.Theria	The wolf is of class Mammalia and sub-class Theria

Fig. 1: Examples of expressions in L

Both $A{:}B$ and $A.B$ imply that B is a way in which A is true. But $A{:}B$ and $A{:}C$ are compatible terms, whereas $A.B$ *excludes* other values of A, such as $A.C$.

1.3 Expressive Impoverishment

At first glance, EL looks like a very impoverished logic indeed, hardly deserving of the name. How can we live without operators for negation and disjunction? Although Exclusion Logic does not contain these operators, it has resources for expressing *some* of what we want to say when we use them.

WhoseMove.Black	WhoseMove.White
Agents:Jack	Agents.Jill
Agents:Jack:Age.37	Agents:Jack:Age.38
Wolf:Class.Mammalia.Theria	Wolf:Class.Gastropoda

Fig. 2: Pairs of incompatible expressions

Incompatibility and Negation in EL When we say $\neg P$, we are saying something which is incompatible with P. But not just any claim which is incompatible with P: $\neg P$ is *the least contentful claim* which is incompatible with P. If we see entailment as a partial ordering such that $P \leq Q$ if P entails Q, then P is the upper bound of all the claims which are incompatible with P. If we sort claims by vagueness, $\neg P$ is the vaguest claim which is incompatible with P.

Suppose someone claims that the flag is red. Now we can make an incompatible claim by saying that the flag is green, or that it is blue, etc. These are all specific information-heavy ways of denying that the flag is red. If we want to say something which is incompatible with the flag being red, but want to give away as little as possible, then we will just say that the flag is not red. EL has the resources for making a specific incompatible claim: given $A.B$, we can make an incompatible claim by saying $A.C$ or $A.D$. It just doesn't have the resources for making the least contentful incompatible claim for any arbitrary complex expression.

We also have the ability in EL to express negation directly using $\rightarrow$. As long as EL contains at least two symbols A and B, we can define an explicit contradiction $\bot$ as $A.A \wedge A.B$. Now define $\neg P$ as $P \rightarrow \bot$. However, although we can express the negation of a simple term such as P or $P.X$, there is no way to express the negation of a *complex* term such as $P \wedge Q$ or $P \rightarrow Q$, because $\rightarrow$ is not recursively embeddable.

Disjunction in EL $P \vee Q$ is the most specific of the claims which is entailed by P and by Q. If we see entailment as a partial ordering, then $P \vee Q$ is the least upper bound of P and Q.

Now certain pairs of expressions in EL have a corresponding least upper bound. For example, the least upper bound of $A.B$ and $A.C$ is A.

But even if they both can serve the same role, A is not the same as $A.B \vee A.C$. The difference is that $A.D \leq A$ for all D, but we do not have $A.D \leq A.B \vee A.C$. Nevertheless, even though they have different inferential consequences, both $A.B \vee A.C$ and A fulfil the role of being the least upper bound of $A.B$ and $A.C$.

1.4 EL and the Sheffer Stroke

By using the exclusion operator ".", EL allows us to make two claims $P.Q$ and $P.R$ which cannot both be true. Now this is reminiscent of the Sheffer stroke, a binary operator $|$, which allows us, by saying $Q|R$, to say that Q and R are incompatible - they cannot both be true.

But there is an important difference: in EL, we can make claims $P.Q$ and $P.R$ which *are* incompatible, but we cannot *say* that they are incompatible within EL itself. EL is explicitatively impoverished. Although we can see, using the semantics of ".", that $P.Q$ and $P.R$ are incompatible, this fact cannot be expressed in EL.

1.5 Semantics

An expression in EL will be interpreted in a lattice of labeled rooted trees. In a labeled rooted tree (hereafter, **LRT**), each vertex is labeled with a symbol from S, and each edge is labeled with either '$*$' or '!'. If the edge from X to Y is labeled with $*$, it means that Y is one of the children of X - but X may have other children also. But if the edge is labeled with !, it means that Y is the *only* child of X.

Definition 2. *An **LRT** is a directed tree with a vertex- and an edge-labeling. Formally, an LRT is a tuple* (V, E, L, M, R) *where:*

- V *is a set of vertices*
- E *is a set of edges*
- L *contains the vertex labels; it is a total function from* V *to the set of symbols* S
- M *contains the edge labels; it is a total function from* E *to* $\{*, !\}$
- $R \in V$ *is the vertex for the root of the tree, where* $L(R) = T$ *(here,* T *is a symbol not occurring in* S *used to label the root of each tree).*

Each element of the tuple yields a corresponding accessor function. Given an LRT X, V_X is the vertices of X, E_X is the edges of X, L_X is the vertex-labels of X, M_X is the edge-labels of X, and R_X is the root vertex of X. Additionally, define E_X^* as the transitive closure of E_X.

Definition 3. *Not all sets of vertices and edges form a valid labeled rooted tree. An LRT is **valid** iff it is :*

- *acyclic: there are no undirected cycles*
- *connected: every vertex can be reached from every other via undirected links*
- *irredundant: no two children of a vertex share the same label*
- *respectful of exclusion: if* $M((x, y)) = !$ *then there are no other edges* $(x, z) \in E$ *for some* $z \neq y$

Valid LRTs In these examples, the $*$ labels have been suppressed. Note that two vertices may share the same label.

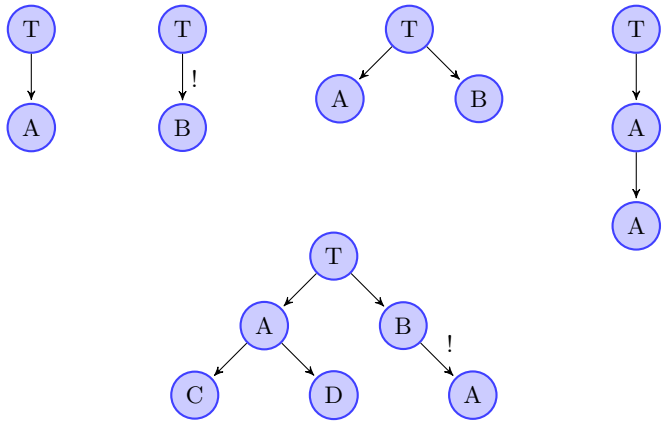


Fig. 3: Valid LRTs

Defining $\leq$ on LRTs We are going to interpret expressions in a lattice of LRTs. So we start with a partial ordering on LRTs. We will say $A \leq B$ if A contains at least as much information as B - if B is a subgraph of A which has the same root.

Now it might seem more natural to define $A \leq B$ if A is a subgraph of B. But we want to preserve the semantic convention that X entails Y iff $[[X]] \leq [[Y]]$, and the corresponding idea that $[[X \wedge Y]]$ is the greatest lower bound of $[[X]]$ and $[[Y]]$.

We will define a partial ordering $\leq$ on LRTs using subgraph isomorphism. We introduce the concept of the signature of a vertex, and say that two vertices from two different graphs are equivalent if they share the same signature.

Definition 4. *The **signature** $s_X(v)$ of a vertex v in LRT X is the ordered list of vertex symbols associated with the path from R_X to v. Because the LRT is a tree, each vertex has only one path from R_X. (Recall that R_X is the root of X).*

Definition 5. *Vertex v in X and vertex v' in Y are equivalent, written $(X, v) \equiv (Y, v')$, iff $s_X(v) = s_Y(v')$. Edges are equivalent if both vertices are equivalent.*

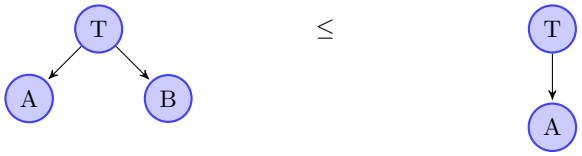
Definition 6. *$A \leq B$ iff every edge in B has a corresponding edge in A, and edge-labels of A are at least as specific as the labels in B.*

$$A \leq B \text{ iff } \forall e \in E_B \; \exists e' \in E_A \; (A, e') \equiv (B, e) \wedge M_A(e') \leq M_B(e)$$

(Recall that E_X is the set of edges of X, and M_X is the set of edge-labels of X). We define $\leq$ over $\{*, !\}$ as the least partial-order on $\{*, !\}$ which satisfies $! \leq *$. Note that it is possible for $A \leq B$ even if A and B have different labels on the same edge, as long as the label on A's edge is more specific than B's label for the corresponding edge.

Examples of $\leq$ Here, again, the $*$ labels have been omitted from the diagram to reduce clutter.

And:



In this example, the LRTs have different labels, but one is more specific than the other:

Proposition 1. $\leq$ *is a partial-ordering.*

Incompatibility Two LRTs are incompatible if they share a path of labeled vertices but diverge at the end of that path, with different labels, and one or both have an exclusive label (!) on the vertex prior to the divergence.

Definition 7. *LRTs A and B are* ***incompatible*** *iff* $\exists(x, y) \in E_A,\ \exists(x', y') \in E_B$ *such that:*

$$\begin{gathered} s_A(x) = s_B(x') \wedge \\ s_A(y) \neq s_B(y') \wedge \\ M_A((x, y)) = ! \vee M_B((x', y')) = ! \end{gathered}$$

Fig. 4: Incompatible LRTs

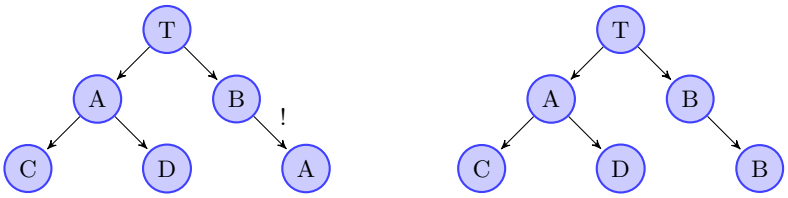
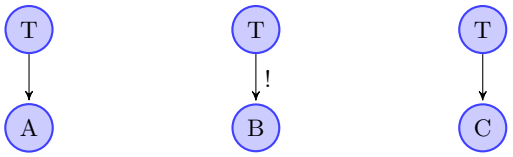


Fig. 5: Incompatible LRTs

Proposition 2. *Incompatibility is symmetric and irreflexive.*

Proposition 3. *Incompatibility is not transitive.*

Proof. The simplest example which shows this is:



(a) (b) (c)

Fig. 6: Incompatibility is not transitive

Here, (a) is incompatible with (b) and (b) is incompatible with (c), but (a) is not incompatible with (c).

LRTs form a lattice We will use LRTs to model expressions in EL, and $\sqcap$ (greatest-lower-bound) to model conjunctions of expressions. So we need some sort of LRT to model incompatible conjunctions of expressions. We add $\bot$ to our set of LRTs to model incompatible conjunctions and stipulate that for all X

$$\bot \leq X$$

Similarly, the lattice needs a top element $\top$ such that for all X, $X \leq \top$. We define $\top$ as

$$(\{1\}, \{\}, \{(1, T)\}, \{\}, 1)$$

Now, with these elements in place, we are ready to define greatest lower bound.

Definition 8. *For any LRTs X and Y, define $X \sqcap Y$ as $\bot$ if X and Y are incompatible; otherwise:*

$$(V_X \cup V_{Y'}, E_X \cup E_{Y'}, L_X \cup L_{Y'}, min_{\leq}(M_X \cup M_{Y'}), R_X)$$

where

$$min_{\leq}(X) = \{(E, m) \in X | m =! \vee (E, !) \notin X\}$$

and Y' is a renaming of the vertices of Y such that $s_X(v) = s_{Y'}(v')$ iff $v = v'$.

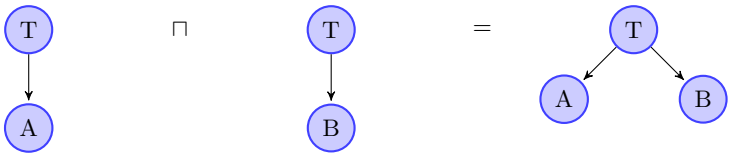


Fig. 7: Example of $\sqcap$

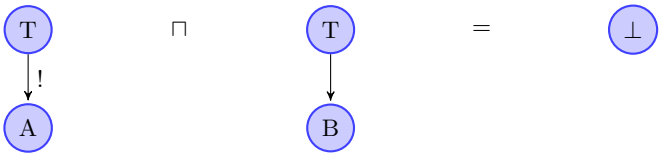


Fig. 8: Example of $\sqcap$

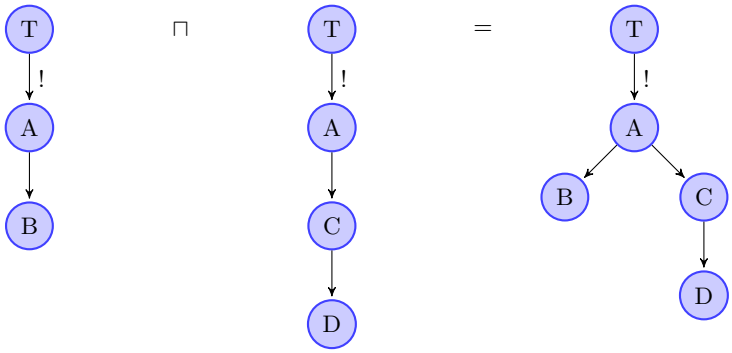


Fig. 9: Example of ⊓

Proposition 4. *$X \sqcap Y$ as defined is indeed the greatest lower bound.*

Definition 9. *For any LRTs X and Y, the least upper bound $X \sqcup Y$ is*

$$(V_X \cap V'_Y, E_X \cap E'_Y, L_X \cap L'_Y, max_{\leq}(M_X, M_Y), R_X)$$

where

$$\begin{aligned} max_{\leq}(X,Y) = \{(E,!) \mid (E,!) \in X \wedge (E,!) \in Y\} \cup \\ \{(E,*) \mid ((E,*) \in X \wedge (E,*) \in Y) \\ \vee ((E,*) \in X \wedge (E,!) \in Y) \\ \vee ((E,!) \in X \wedge (E,*) \in Y)\} \end{aligned}$$

and Y' is a renaming of the vertices of Y, as before.

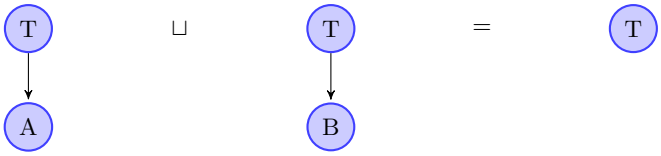


Fig. 10: Example of ⊔

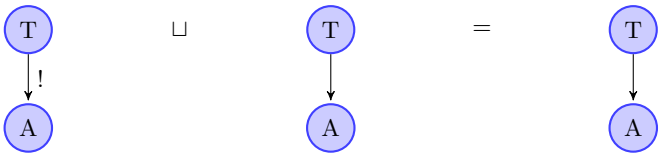


Fig. 11: Example of ⊔

Proposition 5. *Given a set P of objects, the partial order $(LRT(P) \cup \bot, \leq)$ is a bounded lattice with top $\top$, bottom $\bot$, greatest lower bound $\sqcap$ and least upper bound $\sqcup$.*

Defining satisfaction of an expression by an LRT

Definition 10. *We define the conditions for an expression e to be true in an LRT X, written $\models_X e$.*

- $\models_\bot E$, *for all expressions* E
- $\models_X E$ *iff* $Sat(X, R_X, *, E)$

The satisfaction of a term E in a model X, given a vertex v and a label L, written $Sat(X, v, L, E)$, is defined as:

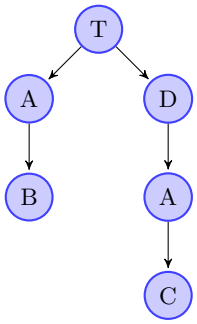
$$\begin{aligned}
Sat(X, v, L, S) \;\; &\textit{iff} \;\; \exists v' : (v, v') \in E_X \\
&\quad\; L_X(v') = S \textit{ and} \\
&\quad\; M_X(v, v') \leq L \\
Sat(X, v, L, S.E) \;\; &\textit{iff} \;\; \exists v' : (v, v') \in E_X \\
&\quad\; L_X(v') = S \\
&\quad\; M_X(v, v') \leq L \textit{ and} \\
&\quad\; Sat(X, v', !, E) \\
Sat(X, v, L, S{:}E) \;\; &\textit{iff} \;\; \exists v' : (v, v') \in E_X \\
&\quad\; L_X(v') = S \textit{ and} \\
&\quad\; M_X(v, v') \leq L \textit{ and} \\
&\quad\; Sat(X, v', *, E) \\
Sat(X, v, L, A \wedge B) \;\; &\textit{iff} \;\; Sat(X, v, L, A) \textit{ and } Sat(X, v, L, B) \\
Sat(X, v, L, A \to B) \;\; &\textit{iff} \;\; (\forall v' \mid (v, v') \in E_X^*) \\
&\quad\; \neg Sat(X, v', L, A) \textit{ or} \\
&\quad\; Sat(X, v', L, B)
\end{aligned}$$

Recall that E_X^* is the transitive closure of E_X.

The conditional of EL is stricter than material implication An implication $p \to q$ is only true at a vertex in a LRT if the material implication holds at *every* vertex below. (In this respect, $\to$ resembles Lewis' strict implication in that its interpretation requires quantifying over a set of points, rather than being interpreted at just one point). The following is not valid:

$$A.B \not\models A.C \to Z$$

The following model satisfies $A.B$ but does not satisfy $A.C \to Z$:



In a sense, implication in EL is *intermediate* between material implication and strict implication. With material implication, we only look at *one* piece of information to determine the truth of the conditional. With strict implication, we look at information from lots of different worlds to determine the truth of the conditional. But in EL, we look at lots of pieces of information from one world.

Validity and Entailment We define validity and entailment in the usual way.

Definition 11. *Define Y is valid, written $\models Y$, if Y is true in all models.*

Definition 12. *We say X entails Y, written $X \models Y$, if every model which satisfies the set of judgments X also satisfies Y.*

Valid	Invalid
$X.Y \to X$	$X.Y \to Y$
$X:Y \to X$	$X:Y \to Y$
$X.Y \wedge X.Z \to A$	$X:Y \wedge X:Z \to A$
$X.Y \to X:Y$	$X:Y \to X.Y$

Fig. 12: Some valid and invalid expressions in L

1.6 Decision Procedure

We will use the lattice of LRTs defined above to provide an efficient decision procedure. We will interpret an expression by the $\bigsqcup$ of the LRTs which satisfy it.

Definition 13. *Define $[x]$ as the set of LRTs which satisfy x:*

$$[x] = \{M \mid \models_M x\}$$

Note that there are an infinite number of LRTs which satisfy an expression. Now because the LRTs form a lattice, $[x]$ has a least upper bound $\bigsqcup$. We can use this least upper bound to calculate entailment *directly*.

Proposition 6.

$$\begin{aligned} X \models Y \; \textit{iff} \forall M \;\; &\models_M X \Rightarrow \models_M Y \\ &\textit{iff} \; [X] \subseteq [Y] \\ &\textit{iff} \bigsqcup [X] \leq \bigsqcup [Y] \end{aligned}$$

Next, we will provide a direct way of computing $\bigsqcup[X]$. This will give us a direct and efficient way of determining if Y is entailed by X: look to see if $\bigsqcup[X] \leq \bigsqcup[Y]$. Instead of having to verify *all* the models of X to see if Y is true, we just look in one canonical model.

Computing $\bigsqcup[X]$ directly We will define a particular model $m(X)$, and then show that m is indeed the minimal model: $m(X) = \bigsqcup[X]$. Note that the expressions in L can be divided into two groups: the conjunctions and the implications.

Definition 14. *First we will define $m(X)$ for the fragment of L that doesn't include implications.*

- $m(T) = \top = (\{1\}, \{\}, \{(1, T)\}, \{\}, 1)$
- $m(A \wedge B) = m(A) \sqcap m(B)$
- $m(A{:}B) = (V_{m(A)} \cup \{v\}, E_{m(A)} \cup \{(v', v)\}, L_{m(A)} \cup (v, B), M_{m(A)} \cup \{((v', v), *)\})$
- $m(A.B) = (V_{m(A)} \cup \{v\}, E_{m(A)} \cup \{(v', v)\}, L_{m(A)} \cup (v, B), M_{m(A)} \cup \{((v', v), !)\})$

Here, v is a new vertex not occurring in $m(A)$ which is used to interpret B, and v' is the vertex which $m(A)$ uses to interpret A i.e. the v' such that $s_{m(A)}(v') = symbols(A)$. Here, $symbols(A)$ is the ordered-list of symbols appearing in A. E.g. $symbols(A : B.C) = (A, B, C)$.

Definition 15. *We extend m from individual expressions to sets of expressions:*

$$m(X) = \sqcap\{m(e) \mid e \in X\}$$

Definition 16. *We extend m to include implications. First, divide a set X of expressions into the conjunctions A and the implications G. We will interpret G as a function, and interpret $m(G, A)$ as the result of repeatedly applying $m(G)$ to $m(A)$. Define $m(X \to Y)$ as the pair of LRTs $(m(X), m(Y))$. Next we will define the evaluation e of a pair (A, B) of LRTs to an LRT. The evaluation function $\boldsymbol{e}$ has signature $LRT \times LRT \to LRT \to LRT$ and is defined as:*

$$e(A, B)(X) = \begin{cases} B \sqcap X & if X \leq A \\ X & otherwise \end{cases}$$

Definition 17. *We extend e to operate on sets of pairs in the natural way: Define* **e'** *as:*

$$e' \ : \ 2^{LRT \times LRT} \to LRT \to LRT$$
$$e' \ G \ X = \sqcap\{e(h, X) \mid h \in G\}$$

Definition 18. *We define $m(G, A)$ to be the least-specific model which is closed under the application of e' on $m(G)$ over $m(A)$:*

$$m(G, A) = \bigsqcup\{X | X \leq e'(m(G), m(A)) \wedge e'(m(G), m(X)) = m(X)\}$$

Proposition 7. *$m(X) = \bigsqcup[X]$ for all sets of expressions X.*

The Complexity of the Decision Procedure We determine whether $G, A \models X$ by checking whether $m(G, A) \leq m(X)$. Assume G is a set of judgments of size n, and A and X are individual conjunctive judgments. Assume the max number of conjunctions in G, A, X is m, and the max complexity of any conjunct is k. Repeatedly applying $\lambda z.e'(m(G), z)$ to $m(A)$ takes of order $n^3m^2k^2$ operations. Determining whether $m(G, A) \leq m(X)$ involves comparing each vertex of $m(X)$ with every vertex in $m(G, A)$. The size of $m(G, A)$ is bounded by $(n + 1)mk$, hence computing $\leq$ is of order nm^2k^2 operations. So, as we would expect, the restricted structure of EL means that testing entailment is polynomial-time and practically efficient.

2 Using Exclusion Logic as a Deontic Logic

Deontic Exclusion Logic, hereafter **DEL**, is a particular application of Exclusion Logic.

2.1 Motivation

The design of DEL was driven by the desire to avoid some of the unintuitive consequences of Standard Deontic Logic (hereafter **SDL**). SDL is a reinterpretation of the modal logic $K + D$ in which $\Box P$ is interpreted as "It is obligatory that P". The syntax of SDL is given by:

$$L ::= P \mid \Box L \text{ where P is the set of all propositional formulae}$$

SDL is interpreted in the usual Kripke semantics with a serial accessibility relation. So the following are valid:

$$\models \Box(p \vee \neg p)$$
$$\models \Box(p \to p)$$

We are obligated to make it the case that tautologies hold! This seems a heavy burden indeed. Further, in SDL we have the entailment:

$$\Box(p) \models \Box(p \vee q)$$

This hold for any proposition q, no matter how unsavoury!

The problem here is at the root, with the very sentences that SDL allows as syntactically acceptable. Just because a formal language accepts a certain sentence, it doesn't mean that we can do anything with it - that we can make it mean anything. When the $\Box$ operator is interpreted deontically, it is not clear what sense to make of $\Box(p \vee q)$ or $\Box(p \rightarrow q)$. These are not formalized versions of expressions which are antecedently intelligible. It is rather that they are forced on us by the syntactic machinery of SDL, and we don't know what to do with them.

> Philosophers are often like little children who scribble some marks on a piece of paper and then ask the grown-up "What does this mean?" [Wittgenstein, Culture and Value, p. 17]

Von Wright originally had a formalism which was much more syntactically restricted. In this earlier version, the deontic operator was applied to *actions*, not to sentences. This meant that it was impossible to express $\Box(p \vee q)$, $\Box(p \rightarrow q)$, or $\Box(\Box(p))$. DEL, like von Wright's original deontic language, is highly syntactically restricted in order to avoid problem cases where syntax outstrips understanding.

2.2 Syntax

Given a set S of symbols, the expressions in DEL are defined as the terms G in:

$$X ::= S \mid S{:}X \mid S.X \mid \Box X$$
$$E ::= X \mid E \wedge E$$
$$G ::= E \mid E \rightarrow E$$

In DEL, like SDL, the intended meaning of $\Box P$ is that P should be the case.

Well-Formed	Ill-Formed
$\Box A \ \wedge \Box B$	$\Box(A \wedge B)$
$A \rightarrow \Box B$	$\Box(A \rightarrow B)$
$\Box\Box A$	

Fig. 13: Well-formed and ill-formed expressions in DEL

Note that DEL's operators are carefully stratified. Unlike SDL, DEL does not tolerate $\Box$ outside a conditional or conjunction. The reason for these restrictions is that in DEL, $\Box P$ is just *syntactic sugar* for $Ob : P$. Here, Ob is just a constant of EL, suggestively chosen to evoke the intended interpretation of obligation. We will see in the next section why it is permissible to interpret Ob deontically.

2.3 Interpreting □: DEL as a Minimal Deontic Logic

For a logic to qualify as a deontic logic, it must respect our intuitions about how the Obligatory operator behaves. These intuitions include, at the very least:

1. Closure under strict implication. Any model in which P strictly-implies Q and $\Box P$ must also satisfy $\Box Q$
2. No incompatible deontic judgments. If P and Q are incompatible propositions, there is no model which satisfies $\Box P \wedge \Box Q$
3. What ought to be the case does not collapse into what is the case. Deontic judgments can remain unsatisfied. $\Box P \not\models P$

In SDL, these requirements are satisfied because the accessibility relation is serial but not reflexive.

In DEL, $\Box P$ is just *syntactic sugar* for $Ob : P$. There are no changes whatsoever to the semantics to incorporate this syntactic extension. But the semantics of EL already respect the three minimal requirements on a deontic logic listed above:

1. Closure under strict implication. In EL, the satisfaction condition for $\rightarrow$ means that $P \rightarrow Q, \Box P \models \Box Q$
2. No incompatible deontic judgments. In EL, the satisfaction condition for $P \wedge Q$ is defined so that if P and Q are incompatible propositions, there is no model which satisfies $\Box P \wedge \Box Q$
3. What ought to be the case does not collapse into what is the case. Deontic judgments can remain unsatisfied. In EL, $\Box P \not\models P$

Now these conditions allow us to interpret $\Box P$ as "It is obligatory that P", but they do not *force* us to interpret it that way. Just as there are other interpretations of $K + D$ in which $\Box$ is interpreted in some way other than by a deontic operator, just so there are other interpretations of DEL in which $\Box$ is interpreted otherwise. But satisfying the conditions above means that $\Box$ *can* be interpreted deontically. Hence DEL is a minimal deontic logic.

2.4 Comparison with SDL

The deontic logic literature contains many types of case which SDL finds problematic for one reason or another. We will briefly review some of these cases in order to see how DEL handles them.

Example 1 (Tautologies are obligatory). Because SDL is a normal modal logic, if P is a tautology, then $\models \Box P$. DEL does not have this bizarre consequence. The only tautologies in DEL are implications (e.g. $A.B \rightarrow A$). But the stratified syntax of DEL restricts the $\Box$ operator from being applied to implications. So there are no tautologies in DEL which can be embedded inside $\Box$.

Example 2 (Ross' paradox). In SDL, the following holds:

$$\Box P \models \Box(P \vee Q)$$

Let $P =$ "You mail the letter" and $Q =$ "You burn it". Then it is an unfortunate consequence that if you ought to mail the letter, then you ought to mail it or burn it.

DEL handles this case in the simplest way, by disallowing any boolean operations within the scope of the deontic operator. But *even if* we added $\vee$ to DEL, it *still* wouldn't allow $\Box(P \vee Q)$, because the operators in DEL are *carefully stratified*: the deontic operator only applies to simple terms, not to complex expressions involving the logical operators.

Example 3 (Chisholm's puzzle). Chisholm's puzzle involves the following claims:

1. Jones should go to the assistance of his neighbors.
2. If Jones goes, then he should tell them he is coming
3. If Jones doesn't go, then he shouldn't tell them he is coming
4. Jones doesn't go to assist his neighbors

Intuitively, the four claims are consistent and independent. The problem for SDL stems from how to interpret (2) as to preserve both consistency and independence. If we interpret it as $\Box(P \to Q)$, then we can derive both that he should and shouldn't go to the assistance of his neighbors. But if, to take the other horn of the dilemma, we interpret it as $P \to \Box Q$, then as the $\to$ of SDL is material implication, (2) follows immediately from (4). This alternative interpretation seems more natural, but as long as we interpret $\to$ as material implication, we have lost our original intuition that the four propositions are logically independent.

DEL handles this dilemma naturally. The four claims are represented as follows:

1. $\top \to \Box\, Go.True$
2. $Go.True \to \Box\, Tell.True$
3. $Go.False \to \Box\, Tell.False$
4. $Go.False$

Here, we are using $P.True$ and $P.False$ as ways of asserting P and $\neg P$ respectively.

Of the two interpretations of (2) available to SDL, only one is available to DEL because DEL's syntax is stratified to prevent deontic operators having wider scope than logical operators. $\Box(P \to Q)$ is not allowed, so DEL uses $P \to \Box Q$. But implication in DEL is stronger than material implication. Recall that $\to$ is interpreted in DEL as:

$$\begin{aligned} Sat(X, v, L, A \to B) \text{ iff } & (\forall v' \mid (v, v') \in E_X^*) \\ & \neg Sat(X, v', L, A) \text{ or} \\ & Sat(X, v', L, B) \end{aligned}$$

With this stronger interpretation of implication, $A.B \not\models A.C \rightarrow Z$, and the troubling inference from (4) to (2) does not go through.

It follows from these claims that both $\Box\Box\, Tell.True$ and $\Box\, Tell.False$. These claims are compatible. $\Box\Box\, Tell.True$ is the important conclusion that *ideally*, if events had unfolded as they should have done, then Jones should have told his neighbors.

2.5 Comparison with von Wright and Castañeda

In von Wright's original deontic logic [9], the deontic operator is applied to action-types, not propositions. This meant his operator could not be iterated, and could not be applied to complex logical sentences (because the operator does not apply to *sentences* at all). Castañeda proposed a similar syntactically restricted approach [5], distinguishing propositions from practitions.

	$\Box\Box P$	$\Box(P \rightarrow Q)$
SDL	Yes	Yes
DEL	Yes	No
von Wright	No	No

Fig. 14: Syntactic Restrictions

DEL allows iterated application of $\Box$, but does not allow $\Box$ to be applied to conditionals. In this respect, it occupies a respectable middle-ground between the confining syntactic restrictions of von Wright's system, and the anything-goes syntactic free-for-all of SDL.

3 Computational Implementation

DEL has been used to power a multi-agent simulation. A number of different social practices were modeled in DEL, including a turn-taking game, manifesting social status, queuing up, and the notion of taboo activity. In these initial experiments, DEL was found to be an expressive and intuitive language for representing social practices declaratively.

4 Summary

This paper has introduced yet another formal language for the representation of norms. One thing that distinguishes this particular formalism is its simplicity: its eschewal of negation or disjunction means that it has an efficient decision procedure.

References

1. Alchourron, C., and D.Makinson: Hierarchies of Regulations and their Logic. New Studies in Deontic Logic. D. Reidel, Dordrecht, pp. 125-48 (1981)
2. Boella, G., van der Torre, L.: Permissions and Obligations in Hierarchical Normative Systems. Proc. of ICAIL (2003)
3. Brandom, R.: Making It Explicit. Harvard University Press (1994)
4. Brandom, R.: Between Saying and Doing. Oxford University Press (2008)
5. Castañeda, H.: The Paradoxes of Deontic Logic. New Studies in Deontic Logic. D Reidel (1981)
6. Evans, R.: The Logical Form of Status-Function Declarations. Etica & Politica, vol.XI, pp.203-259 (2009)
7. Makinson, D., van der Torre, L.: Input/Output Logics. Journal of Philosophical Logic, vol.29, pp. 383-408 (2000)
8. Makinson, D., van der Torre, L.: Permission from an Input/Output Perspective. Journal of Philosophical Logic, vol.32, pp. 391-416 (2003)
9. von Wright, G.H.: Deontic Logic. Mind, vol.60 (1951)
10. von Wright, G.H.: An Essay in Modal Logic. New York Humanities Press (1953)