Philosophy 244: Modal Logic—Preliminaries

What is modal logic?

Metaphysical answer: It's the logic of "modes." "Modes" are ways in which a condition can hold or obtain or be implemented. Descartes thought all properties were modes either of extension or thought. Triangularity is a way of being extended; reasoning is a mode of thought. The modes in (*alethic*) modal logic are ways of being true. The modes in *deontic* modal logic are ways of being do-able; and so on.

Verbal answer: if propositional logic is the logic of "and", "or", "not", "if-then," modal logic is the logic of "necessarily" and "possibly," "must" and "may."

Historical answer: It's a logic devised in the late 19th and early 20th centuryt in an attempt to deal with the "paradoxes of material implication." E.g., $\neg q \supset (q \supset r)$ is valid but $\neg q$ does not seem to imply that q implies r.

By CI Lewis in his 1910 Harvard dissertation "The Place of Intuition in Knowledge" (advised by Josiah Royce). Lewis credits earlier work of Hugh MacColl's.

Logical answer: Modal logic is the logic of expressions L and M analogous to "necessarily" and "possibly" in (some or all of) the following respects:

1. they're sentential adverbs; if s is a sentence, so are Ls and Ms.
2. one is the "dual" of the other, as $\forall$ is the dual of $\exists$; Ms iff $\neg L \neg s$ and vice versa
3. Ls entails s entails Ms

"A necessarily belongs to all B; B belongs to all C; therefore A necessarily belongs to all C" (Aristotle)

Examples of expressions related in the relevant way.

L	M	$\neg L \neg$
necessarily	possibly	= not-necessarily-not
provable	consistent	= not-provable-not
required	permitted	= not-required-not
known....	for all we know,...	= not-known-not
desirable	acceptable	= not-desirable-not
inevitably	potential;y,...	= not-inevitably-not
probably...	could be that....	= not-probably-not
always	sometimes	= not-always-not
definitely	maybe	= not-definitely-not

"Let this doctrine [of necessary belonging] rest in peace, without giving the least disturbance to its ashes." (Reid)

Not all of these meet all the conditions; the third in particular is often violated. (Obligatory, provable.) But most have got to be met. Operators paired in the ways just vaguely gestured at are called modal operators and symbolized using $\Box$ and $\Diamond$. The book uses L and M and that notation will be used as well, e.g., when we're posting information on the web or communicating via email.

So much for the philosophy. The course will be divided into two parts. First comes *propositional* modal logic: the result of attaching modal operators to ordinary truth-functional logic. Then we'll do *quantified* modal logic: same thing but starting from quantificational logic. Both parts will begin with a quick review of basic facts about the underlying "premodal" languages. So let's remind ourselves how propositional logic works. If you don't already know though this review isn't going to teach you; the point is really just to refresh our memories and get us talking the same logical language.

Topics for today (possibly continuing into next time) are propositional *syntax* – the heart of this being an explanation of what counts as a propositional-calculus sentence – propositional *semantics* – an explanation of how PC sentences are interpreted—and propositional *validity*.

Language of PC (Propositional Calculus)

The language's primitive or undefined symbols are the lower-case letters *p, q,* r,.... with or without numerical subscripts; the connectives $\neg$ and $\vee$; and, for disambiguation

purposes, the parentheses (and). Finite sequences of these primitive symbols are expressions. The well-formed formulas or wffs – intuitively, the sentences – are a subset of these defined by the following formulation rules:

FR1 Every letter is an (atomic) wff.
FR2 If α is a wff, so is $\neg\alpha$.
FR3 If α and β are wffs, so is $(\alpha\vee\beta)$.

Implied closure clause: nothing is a wff that cannot be reached by finitely many applications of these rules. Examples of wffs? &, $\supset$, and $\equiv$ are defined in the usual way. For convenience we allow ourselves to omit outermost brackets.

$\alpha\&\beta =_{df} \neg(\neg\alpha \vee \neg\beta)$.
$\alpha\supset\beta =_{df} \neg\alpha \vee \beta$.
$\alpha\equiv\beta =_{df} (\alpha\supset\beta)$ & $(\beta\supset\alpha)$

Interpretation

Wffs can be thought of as standing (in) for propositions. Every proposition has a truth-value, either true or false. The propositions are closed under a number of proposition-forming operations of which two are negation and disjunction. A proposition's negation is true iff it is false, and vice versa, and the disjunction of two propositions is true iff either of the two is true, otherwise false.

Which propositions does a given wff stand in for? The letters or atomic wffs stand in for any proposition you like. But given a particular assignment of propositions to the atomic wffs, the interpretation of nonatomic wffs is severely constrained, in fact completely determined: $\neg\alpha$ stands in for the negation of what α stands for, and $\alpha\vee\beta$ stands for the disjunction of what α stands for and what β stands for. That is,

IR1 Each atomic wff α stands for an (unspecified) proposition $|\alpha|$.
IR2 If α is a wff, then $|\neg\alpha| =$ the negation of $|\alpha|$
IR3 If α and β are wffs, then $|\alpha\vee\beta| =$ the disjunction of $|\alpha|$ and $|\beta|$.

A key fact about negation and disjunction, considered as operations on propositions, is that the truth-value of a negation is completely determined by that of the proposition negated; likewise for the truth-value of a disjunction vis a vis its disjuncts. This is one way of understanding how and why *sentential* negation and disjunction are truth-functional.

Not every operator is like this. Examples? Counterfactual constructions, attitudinal operators, and of course necessitation. You can't predict the truth-value of the proposition that necessarily BLAH from the truth-value of the proposition that BLAH.

What logical difference does it make whether an operator is or is not truth-functional? Logic is concerned with validity. It wants to identify the sentences α that are true regardless of how its atomic parts are interpreted: regardless of what propositions they are taken to express. This project is way simplified if all of α's connectives are truth-functional. Because then to determine whether α is true on all interpretations, we don't need to randomize over all the (infinitely many!) propositions its atomic parts might express; it's enough to randomize over all the truth-values its atomic parts might have.

In practice then we can *abstract away* from the propositions and focus just on truth-value. Atomic wffs can be thought of as standing not for propositions but truth-values, even if philosophically speaking these are the truth-values of implicitly understood propositions; and negation and conjunction can be understood as operations on truth-values, even if philosophically speaking the negation of a truth-value is the truth-value of a negated proposition. This allows us to replace IR1-3 with ER1-3

ER1 Every atomic α has a truth-value; $[\![\alpha]\!] =_{df}$ 0 or 1.
ER2 $[\![\neg\alpha]\!] =_{df}$ 1-$[\![\alpha]\!]$
ER3 $[\![\alpha\vee\beta]\!] =_{df}$ max($[\![\alpha]\!]$, $[\![\beta]\!]$)

α	β	$\neg\alpha$	$\neg\beta$	$\alpha\vee\beta$	$\alpha\&\beta$	$\alpha\supset\beta$	$\alpha\equiv\beta$
1	1	0	0	1	1	1	1
1	0	0	1	1	0	0	0
0	1	1	0	1	0	1	0
0	0	1	1	0	0	1	1

The information in ER2 and ER3 is often expressed in truth-table form; let's tack on the defined connectives too:

So, negation and disjunction are (as we see it) first and foremost operators on propositions; secondarily operations on truth-values; and thirdly (tertiarily?) syntactical operations. The reason that $\neg$ is called the negation sign is that the result of putting it in front of a wff is a sentence whose truth-value is the semantic negation of that wff's truth-value. The wedge $\vee$ is called the disjunction sign for the same sort of reason. All of that having been said, in practice we'll use the words "negation" and "disjunction" ambiguously to stand sometimes for the symbol and associated syntactic operation, sometimes for the semantic operation on truth-values, and sometimes for the semantic operation on propositions, with no attempt to enforce any priority among these operations.

Propositional Validity

Remembering that the letters or propositional variables of PC stand for arbitrary propositions, a wff is (intuitively, informally) *valid* iff every way of putting propositions in for its component letters results in a truth. The reason that $(p\&q)\supset(p\supset q)$ is valid is that if you consider all of propositions arrived at by putting propositions in for p and q, you find that every single one of them is true. Again, that's the intuitive, informal idea. Given that all the connectives (operators) in a PC wff are truth-functional, we may as well define validity not as truth under all assignments of propositions to p, q, etc., but rather truth under all assignments of truth-values to p, q, etc. There is also the opposite property of being *false* under all assignments of truth-values to variables. A wff with the first property is called PC-valid or a tautology, one with the second property is PC-antivalid or unsatisfiable. Most wffs of course fall somewhere between.

E.g.,

if goats eat cans and bottles, they eat bottles if they eat cans,
if bees buzz and sting, they sting if they buzz
.....

This definition of validity, although perfectly accurate, does not adapt as naturally as we might like to the modal case. So let's give in addition a second definition in terms of a certain sort of game, the PC-game. The PC-game is like solitaire; it has only one player. (Have no fear, lots of players will be needed when the game is elaborated to accommodate modality.) Here's how it goes:

The player has a sheet of paper on which a number of letters (e.g., p, r, s,...) are written. Player and paper together are called a *PC-setting*. PC-settings differ only in which letters are written on the paper. We now proceed to call out to the player a series of PC wffs, subject to the following requirement: we never call out a wff until all its truth-functional components have been called, e.g., before we call a disjunction we must call each of its disjuncts, and if one of the disjuncts is a negation, before we call it we must call its negatum. The player's instructions are these:

1. If a letter is called, raise your hand iff that letter is on the sheet.
2. If $\neg\alpha$ is called, raise your hand iff you did not raise your hand when α was called.
3. If $\alpha\vee\beta$ is called, raise your hand iff you raised your hand for either α or β.

Say that a wff is *successful* in a setting if the player raises her hand when it is called. (Examples.) Your average wff will be successful in some settings and unsuccessful in others. Our interest though is in the wffs that are successful in every setting: the PC-successful wffs. The alternative definition is now this: a wff is PC-valid iff it is successful in every PC-setting, that is, it is PC-successful.

Using the definitions of &, $\supset$, and $\equiv$, we can easily figure out the rules for conjunctions, conditionals, and equivalences. (Exercise!)

Testing 1: Exhaustive

OK but how do we *test* whether a given wff α is valid? One idea would be to play the PC-game from every setting and see whether the hand always goes up when α is called. But that would take forever; there are infinitely many atomic letters and so infinitely many (in fact uncountably many) settings.

A first step towards containing the problem is to notice that since α is finitely long, it can have only so many letters in it: say *n*. Letters that aren't among these *n* are neither here nor there as far as α's success in a given setting is concerned. Whether they are on the sheet or off it has no effect whatever on α's fate.

It follows that nothing is lost if we limit ourselves to these *n* letters and consider just the settings you get by putting selections of these *n* letters on the sheet. How many is that? 2^n. So, shall we now go ahead and play the game 2^n times to see how α comes out? My mom would say, let's not and say we did. Better to mimic the process of playing 2^n times by writing down representations of the various settings and calculating for ourselves when the hand would have gone up had the game been played.

The example in the book on page 10 is this. Let α be $((p\supset q)\&r)\supset((\neg r\vee p)\supset q)$. This has 3 variables so there are going to be 2^3 or 8 settings. We can represent these various settings as strings of 1s and 0s, 1 for the case where a given letter is on the sheet, 0 for the case where it isn't. The various columns on the right correspond to the various rounds of play; the column marked (1) indicates what happens in the first round, with the various entries in a given column telling you whether the hand went up or stayed down in that round on the associated setting. Eventually we reach a determination of what happened in the last round of each game, when the target sentence α was called. Since the central column consists entirely of 1s, the sentence is PC-valid.

p	*q*	*r*	((*p*	⊃	*q*)	&	*r*)	⊃	((¬	*r*	∨	*p*)	⊃	*q*)
1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
1	1	0	1	1	1	0	0	1	1	0	1	1	1	1
1	0	1	1	0	0	0	1	1	0	1	1	1	1	0
1	0	0	1	1	0	0	0	1	1	0	1	1	1	0
0	1	1	0	1	1	1	1	1	0	1	1	0	0	1
0	1	0	0	1	1	0	0	1	1	0	1	0	1	1
0	0	1	0	0	0	1	1	1	0	1	1	0	0	0
0	0	0	0	1	0	0	0	1	1	0	1	0	1	0
			(1)	(2)	(1)	(3)	(1)	(5)	(2)	(1)	(3)	(1)	(4)	(1)

This is of course the truth-table method for checking validity. An advantage is that it settles not only α's validity but all its logical properties, e.g., whether it is satisfiable or not. If we're willing to settle for information about just validity there's a quicker and slightly more interesting method available.

Testing 2: Targeted

The reductio method. Here we search for a falsifying assignment of truth-values (a game w.r.t. which α is unsuccessful); if it can't be found, α is valid. Explanations can be found on p. 11; I'll just work an example.

((*p*	⊃	*q*)	&	*r*)	⊃	((¬	*r*	∨	*p*)	⊃	*q*)
0	1	0	1	<u>1</u>	0	1	<u>0</u>	1	0	0	0
(9)	(4)	(8)	(2)	(5)	(1)	(11)	(12)	(6)	(10)	(3)	(7)

Each of the two methods gives us a decision procedure for PC: a finite and mechanical method for judging validity. To test this assertion, look at the list the book gives of (what it dogmatically asserts are) valid wffs. What do the two methods say?

24.244 Modal Logic

Spring 2015