Modal logics, temporal and dynamic

Adam Gashlin

Contents

Overview

This report deals with a branch of formal logic called Modal Logic, in which we are concerned with the truth of propositions in different situations, and how those situations relate to each other logically. Example situations include points in time, conceivable worlds, believable knowledge, and states of a computation.

There is a good deal of theoretical mechanism that applies equally well to many kinds of Modal Logic, such as a the unifying Kripke semantics. The essential core of Kripke semantics is a binary relation on the situational "worlds", which defines how propositions in one world may be referenced by propositions in another. This report examines in detail two particular families of modal logic, Temporal Logic and Dynamic Logic.

In temporal logic truth is given for particular points in time. Times are related by "*s* is in the past of *t*" and "*s* is in the future of *t*". We can express statements with reference to some or all times, future or past, such as "*p* or *q* will be true" or "if *r* was always true, *r* is true". These are useful, for instance, for verifying the eventual proper behavior of programs, invariants such as "if the operating system gets a request, the request will be granted".

In dynamic logic, truths hold for states of an executing program, and the relations are the input/output relation of each program. This lets us express such statements as "when program *a* finishes, *p* is true" and "if *p* or *q* are true, then when program *b* finishes, *q* is true". This can be useful for formally specifying correctness properties, such as "if *a* is positive before the loop, *a* is zero after the loop terminates".

Propositional logic

Formal logic

Our concern in a formal logic is to abstract away every detail that is not necessary for preserving the validity of statements. Much reasoning is valid on a formal level: a properly formed argument is assured to be valid by virtue of its form. This abstraction allows logicians to study the structure of logic apart from dealing with the specifics of arguments, and it encourages attention to the essential details of a situation.

Most of the operations of formal logic can be executed mechanically, to varying degrees of effectiveness. There can be great practical value in expressing a problem in the terms of a logic, if there are efficient automatic methods available.

Propositions

A *proposition* is simply a statement which we regard as true or false. An *atomic proposition* is defined to have no internal structure that we are interested in.

Consider the two atomic propositions

- p = "My pants are blue"
- q = "My hair is black"

By calling these atomic, we are ignoring as irrelevant any common details, such as that both of these statements are about me and involve colors. An atomic proposition only has an identity, like the letters p and q here, which lets us know whether propositions can be considered identical.

Connectives

In propositional logic, we reduce statements to atomic propositions, joined by a small family of connectives into larger propositions. If we wanted a proposition that means "my pants are blue and my hair is black", we might introduce a new atomic proposition called s, but this would be a poor choice. For instance, it would allow us to make the statement "s, but not p". This is never true given the earlier definition of p (it is a *contradiction*), but that is not clear from the form of the statement.

Instead of s, we can call this proposition $p \wedge q$. The propositional connective $\wedge$ in the middle is read as "and", and the combined proposition is the statement "both p and q are true".

By using this standard method of connecting existing propositions to name a new one, we get a name that is much more useful. If we know that both proposition p and proposition q are true, we automatically know that proposition $p \wedge q$ is true, without having to look up any details about what these propositions mean. Likewise the contradiction from earlier would be expressed as "$p \wedge q$ but not p", in which the contradiction is immediately apparent.

Evaluation

We haven't said yet what color my hair or pants actually are. There are some statements whose truth doesn't depend on the truth of the atomic propositions, for instance contradictions are always false and their opposite, *tautologies*, are always true. But for everything else, the truth value of a proposition depends on the truth values of the atomic propositions it is made from.

The process of determining the truth value is *evaluation*, and it proceeds according to the structure of a statement. For a statement like the $p \wedge q$ we have been considering, consult the *truth table* to the right.

$\wedge$

φ	ψ	$\varphi \wedge \psi$
T	T	T
T	F	F
F	T	F
F	F	F

The Greek letters φ (phi) and ψ (psi) here are used to indicate any proposition, not just an atomic proposition. In $p \wedge q$, $\varphi = p$ and $\psi = q$. If φ is **T**rue and ψ is **F**alse, $\varphi \wedge \psi$ is **F**alse.

The benefit of this method over saying "$\wedge$ is 'and'" is that the table leaves no room for interpretation. While there may not be many ways of interpreting "and", consider "or", which is written as $\vee$ in propositional logic, and which has the truth table to the right.

$\vee$

φ	ψ	$\varphi \vee \psi$
T	T	T
T	F	T
F	T	T
F	F	F

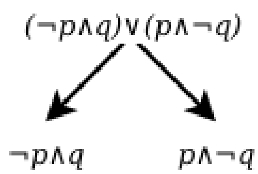
In English it is not unusual for "x or y" to exclude the case where both x and y are true, but the truth table tells us unambiguously that φ **T**rue and ψ **T**rue makes $\varphi \vee \psi$ **T**rue.

Consider a more complicated statement such as $(\neg p \wedge q) \vee (p \wedge \neg q)$. First off, $\neg q$ means "not q", which has the truth table:

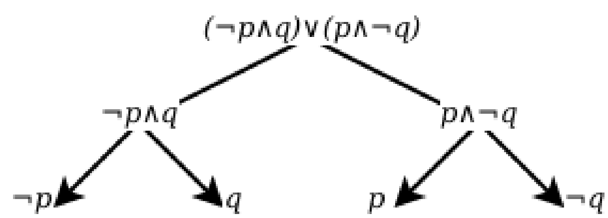
φ	$\neg\varphi$
T	F
F	T

At this point it is helpful to think of each of the atomic propositions p and q as simply something that can be true or false. If we say that p is true and q is false, this forms our *model* for the state of the world, our model is formed from the *valuation* "p is true, q is false". To evaluate $(\neg p \wedge q) \vee (p \wedge \neg q)$, we break it down until we arrive at statements whose truth can be determined directly from the model, then we build back up to the full statement. This can be pictured as building a tree

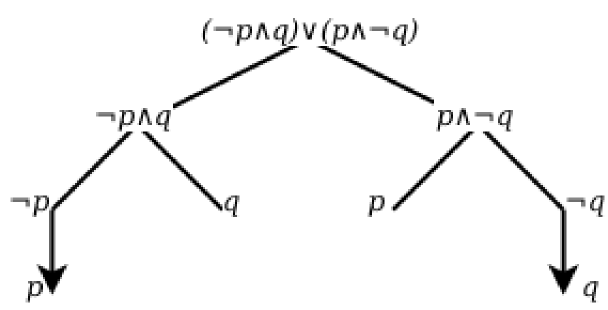
1



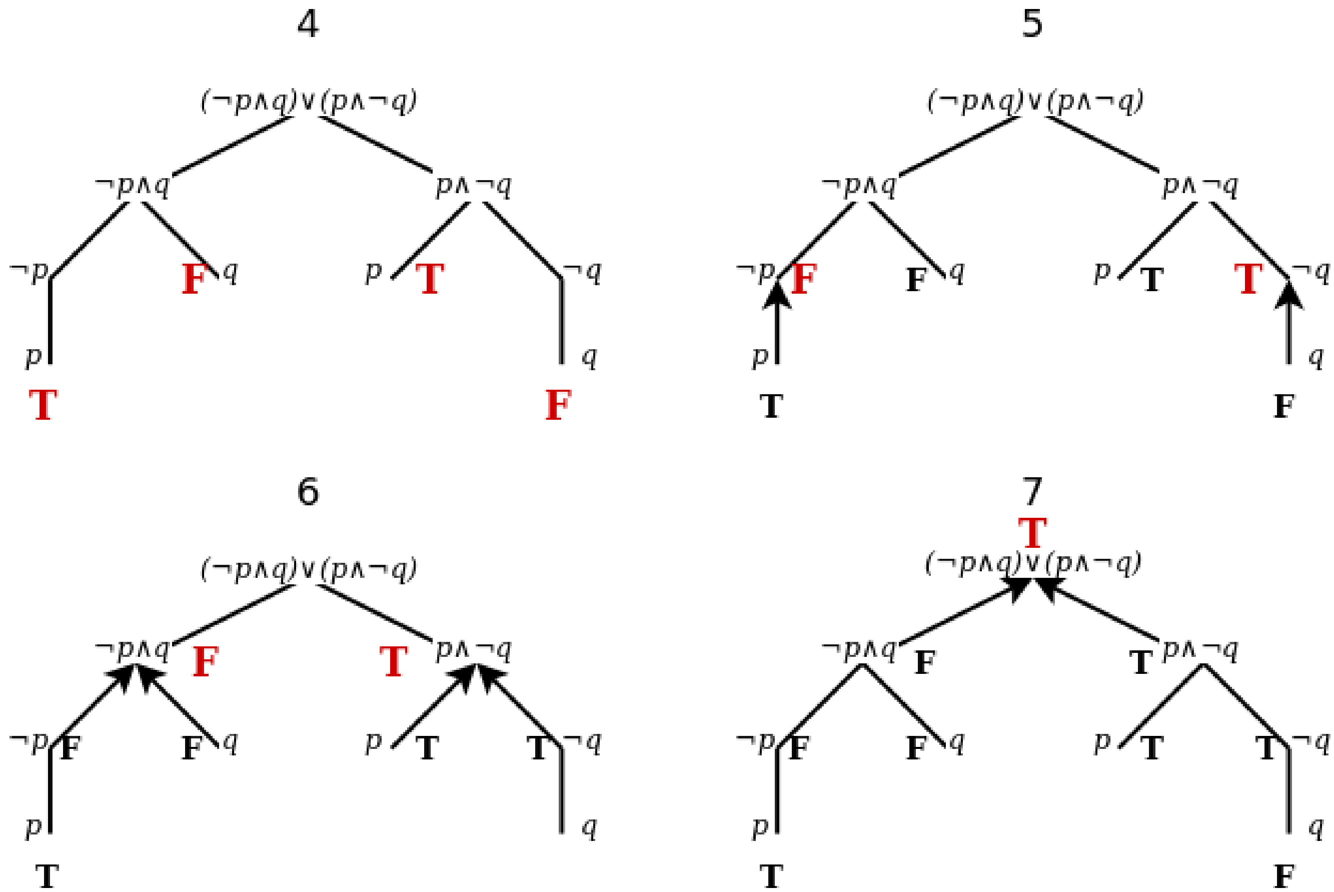
2



3



and filling it with truth values.



Thus for the this model, with the valuation "p is true, q is false", $(\neg p \wedge q) \vee (p \wedge \neg q)$ is true.

Evaluation by definition of $M \models \varphi$

Formally, we treat evaluation as extending the valuation to cover all possible propositions formed from the atomic propositions. If a proposition φ is true in model M, we write $M \models \varphi$.

$M \models p$	if p is a true atomic proposition
$M \models \neg\varphi$	if $M \models \varphi$ is not true
$M \models \varphi \wedge \psi$	if $M \models \varphi$ and $M \models \psi$
$M \models \varphi \vee \psi$	if one or both of $M \models \varphi$ and $M \models \psi$

Note that only one of these rules applies for every possible proposition, so if the left hand side matches the form of a proposition but the condition on the right hand side doesn't hold, then the proposition is false. If $M \models \varphi$ is not true, we instead write $M \not\models \varphi$.

Modal logic

Multiple states

So far everything discussed holds for classical propositional logic as well as modal logic. The first place where modal logic differs is that a model can have multiple *states*, a.k.a. "situations" or "worlds", whereas classical propositional logic considers only one set of truths at a time.

As an example, say we have one state, call it *s*, where the atomic propositions *p* and *q* are both true. We also have a state *t*, where only *p* is true and *q* is false. In modal logic a model *M* has at least two components: the set of states S_M and a valuation V_M. In the example, $S_M = \{s, t\}$, the two states, and V_M is a function that gives the atomic propositions that are true in a specified state: $V_M(s) = \{p, q\}$ and $V_M(t) = \{p\}$.

To show that we are discussing truths in a particular state *s*, we write $M \models_s$ instead of simply $M \models$. In the running example, $M \models_s p$ and $M \models_s q$, as well as $M \models_t p$. However it is not the case that $M \models_t q$, so we write $M \not\models_t q$. We can also use the earlier definition of $\models$ to extend $\models_s$ and $\models_t$ to larger propositions, for instance $M \models_s p \wedge q$ and $M \models_t p \wedge \neg q$.

Often we want to reason without knowing what state is the "real world". If we find a proposition that is true across all states, we can use it without reference to a particular state. If this is so we say the proposition is *true in the model*, and we write $M \models \varphi$ without a subscript. In the example we have $M \models p$ and $M \models q \vee p$, for instance.

Referring to other states

The real power of modal logic comes from being able to make reference to other states. For this purpose we add two symbols to our proposition language: $\Box$ ("box") and $\Diamond$, ("diamond"). These are used to prefix other propositions to modify their interpretation. $\Box\varphi$ means that in all related states (to be defined in the next section), φ is true, whereas $\Diamond\varphi$ asserts that there is at least one related state where φ is true.

Note that $\Box\varphi$ and $\Diamond\varphi$ are only about related states, they make no reference to what is true in the current state. To claim that φ is true in the current state and in all related states, use the proposition $\varphi \wedge \Box\varphi$. Also note that $\Box\varphi$ can be true even if there are no states where φ is true; this is the case when there are no related states.

We can formally define $\Box$ and $\Diamond$ by adding the following rules to our definition of $\models$:

$$\begin{array}{ll} M \models_s \Box\varphi & \text{if for all states } t \text{ where } sR_Mt,\ M \models_t \varphi \\ M \models_s \Diamond\varphi & \text{if there is a state } t \text{ such that } sR_Mt \text{ and } M \models_t \varphi \end{array}$$

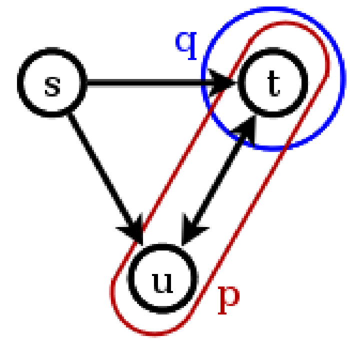
sR_Mt means that *s* is related to *t* by relation R_M, which we will explain in the next section.

Related states

The meaning of $\Box$ and $\Diamond$ in a given model is specified by a binary relation between pairs of states. A *binary relation* is simply a set of ordered pairs of objects. The term "binary" here means that the relation deals with two objects at a time, i.e. it is about pairs. If a relation *R* contains a pair of states, then those states are considered to be related by *R*. For example the relation $R = \{(s, t), (s, u), (t, u), (u, t)\}$ means that *s* is related to *t* by *R*, which we write as *sRt*. The converse is not true, however: it is not the case that *tRs* (as there is no pair (t, s)). Both *tRu* and *uRt* are true.

One way to understand the meaning of a relation in a model is that is describes the set of states that are accessible, by $\Box$ and $\Diamond$, from a given state. $M \models_s \Box q$ only refers to the truth of *q* in those states *t* that are related to *s* by sR_Mt.
More on this in the examples below.

An example



For a simple example (based on Harel et al. exercise 5.1), consider a model with the states S_M, relation R_M, and truth values determined by V_M:
$S_M = \{s, t, u\}$,
$R_M = \{(s, t), (s, u), (t, u), (u, t)\}$,
$V_M(s) = \{\}, V_M(t) = \{p, q\}$,
$V_M(u) = \{p\}$.

This is depicted in the diagram on the right, where circled letters are states, arrows show relations, and colored shapes enclose states where the like-colored propositions are true.

Here, we have $M \models_x \Box p$ for any state *x*, as all states *y* that *x* is related to (i.e. that it has an arrow from x to y) have $M \models_y p$. Therefore we can write $M \models \Box p$.

We also have $M \models_s \Diamond q$ and $M \models_u \Diamond q$, but not $M \models_t \Diamond q$.

$M \models_t \Diamond \Diamond q$ is true, as there is some state (*u*) related from *t* that relates to some state (*t* again) where *q* is true.

Frames

A proposition can be true in one state of a model, or it can be true across all states in the model. Then there are tautologies such as $p \vee \neg p$, which are true regardless of model, as they are true regardless of the truth values of the atomic propositions. In between propositions that are and that are true in a model, there is another level of propositions that are *valid in a frame*.

1. φ is true in a state *s* in a model *M*, $M \models_s \varphi$
2. φ is true in a model *M*, true in all states of the model, $M \models \varphi$
3. φ is valid in a frame *F*, true on all models on a frame, $F \models \varphi$
4. φ is a tautology, valid regardless of frame, $\models \varphi$

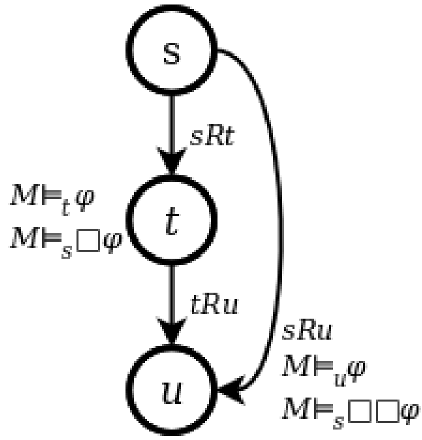
A *frame* consists of just the set of states S_M and the relation R_M used to construct the model, that is it does not specify what the truth values actually are in the states. The propositions that are valid on a frame thus depend on the properties of the relation. Some examples follow (from Goldblatt, p. 12, theorem 1.12).

Reflexivity
: If a relation says that a state is always related to itself (*sRs* for all states *s*, the relation is called *reflexive*. In any frame that has a reflexive relation, all propositions of the form $\Box\varphi \to \varphi$ will be valid. If all related states have φ true, then this state must have φ true as well, since it is related to itself.

Symmetry
: In a symmetric relation *R*, if *sRt* then *tRs*, so all relations go both directions. In a frame with a symmetric relation, all propositions of the form $\varphi \to \Box \Diamond \varphi$ are true. That is, if φ is true at *s*, then in every state *t* related to *s*, there is some related state *u* where φ is true. To see why this is true, *u* can just be *s* since both *sRt* and *tRs*.

Transitivity
: A transitive relation is always "inherited" through a middle state. If *sRt* and *tRu*, then, if *R* is a transitive relation, it is also the case that *sRu*. This allows the middle state to be cut out so that the

relation applies directly between two states that are otherwise known to be second-degree related.



In a frame with a transitive relation, all propositions with the form $\Box\varphi \rightarrow \Box\Box\varphi$ are valid. This is because, as shown on the right, all states that are second-degree related (such as *u*) are also first-degree related (because of transitivity), thus if all first-degree related states have φ (i.e. $\Box\varphi$), then all second-degree related states have φ (i.e. $\Box\Box\varphi$).

Seriality

A serial relation is usually on an infinite set, like a set of points in time that we assume will go on for ever; seriality is the property that any state always has a related state. In a frames with a serial relation, all propositions with the form $\Box\varphi \rightarrow \Diamond\varphi$ are valid. This is simply the statement that we don't have any dead ends where there are no related states, where $\Box\varphi$ can be true without φ actually being true anywhere (because "all related states" is nothing).

Bibliography

- Patrick Blackburn, Maarten de Rijke, and Yde Venema. 2001. Modal Logic. Cambridge University Press.
- Robert Goldblatt. 1987. Logics of Time and Computation. Center for Study of Lang. and Info.
- David Harel, Jerzy Tiuryn, and Dexter Kozen. 2000. Dynamic Logic. MIT Press,.