Expressivity and Inference in Hybrid Logic

Patrick Blackburn

Section of Philosophy and Science Studies, IKH, Roskilde University, Denmark

The Second Tsinghua Logic Summer School

June 27 – July 3, 2022, Beijing, China

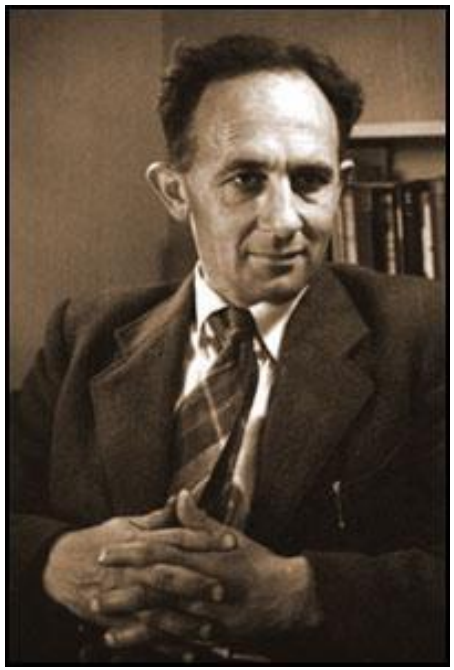
Goals of the course

This course introduces hybrid logic, a form of modal logic in which it is possible to name worlds (or times, or computational states, or people, or whatever it is that the elements of Kripke models are taken to represent).

The course has three major goals:

- First, to make the ideas and intuitions that have guided the development of hybrid logic clear.
- Second, to prove some key results about hybrid inference and completeness.
- Third, to examine the ways in which hybrid logic is related to first-order logic.

But also: a little history...



For hybrid logic, history starts with Arthur Prior (1949 – 1969). Prior is probably best known as the inventor of tense logic, the logic of future and past, which evolved into the various temporal logics used in modern computer science — though some of you may have come across papers like *The Runabout Inference Ticket* or *Thank Goodness That's Over*.

But he also invented hybrid logic — something that was long overlooked.

Course outline

- Lecture 1: From modal to hybrid logic
- Lecture 2: Hybrid inference
- Lecture 3: Hybrid completeness
- Lecture 4: Adding downarrow
- Lecture 5: Arthur Prior and strong hybrid languages

Format

Every day three sessions of about 45 minutes.

- 2×45 minutes of lecture (with a 5-10 minute break after each.
- 1×45 minutes for questions and answers, discussion of homework exercise, revision of important concepts . . . .

Three homework assignments: Monday, Tuesday, and Wednesday. Answers to each assignment will be posted online the following day.

There is also an exam. The questions are similar to those on the homework exercises. The exam will be posted on Friday.

Teaching assistant: Chen Qian.

Today: From modal to hybrid logic

In today's **lecture** **(2×45 minutes)** I will present:

- Orthodox modal logic — from an Amsterdam perspective.
- A problem with orthodox modal logic.
- Fixing this problem with basic hybrid logic.
- Technical themes: The standard translation, frame definability, and bisimulation.
- Conceptual theme: What hybrid logic is, and why it is modal.

In today's **tutorial session** **(1×45 minutes)** we will discuss the completeness result for basic modal logic

What is modal logic?

Slogan 1: Modal languages are simple yet expressive languages for talking about relational structures.

Slogan 2: Modal languages provide an internal, local perspective on relational structures.

Slogan 3: Modal languages are not isolated formal systems.

These slogans pretty much sum up the Amsterdam perspective on modal logic.

Slogans from: Preface to *Modal Logic*, by Blackburn, de Rijke and Venema, Cambridge University Press, 2001

Propositional modal logic

Given propositional symbols PROP $= \{\mathrm{p}, \mathrm{q}, \mathrm{r}, \ldots\}$, and modality symbols MOD $= \{m, m', m'', \ldots\}$ the basic modal language (over PROP and MOD) is defined as follows:

$$\begin{aligned} \mathrm{WFF} \quad := \quad & \top \mid \bot \mid \mathrm{p} \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi \\ & \mid \langle m \rangle \varphi \mid [m]\varphi \end{aligned}$$

If there is just one modality symbol in the language, we usually write $\Diamond$ and $\Box$ for its diamond and box forms.

$\Box\varphi$ can be regarded as shorthand for $\neg\Diamond\neg\varphi$. And $\Diamond$ can be regarded as shorthand for $\neg\Box\neg\varphi$. That is, $\Box$ and $\Diamond$ are dual operators.

Kripke models

- A Kripke model $\mathcal{M}$ is a triple $(W, \mathcal{R}, V)$, where:
  - W is a non-empty set, whose elements can be thought of as possible worlds, or epistemic states, or times, or states in a transition system, or geometrical points, or people standing in various relationships, or indeed, pretty much anything you like.
  - $\mathcal{R}$ is a collection of binary relations R^m on W, one for each element of MOD. That is, $R^m \subseteq W \times W$, for all $m \in \mathrm{MOD}$.
  - V is a valuation assigning subsets of W to propositional symbols. That is, $V(p) \subseteq W$ for all $p \in \mathrm{PROP}$.
- The component $(W, \mathcal{R})$ is traditionally called a frame.
- When working with a single modality, we write models $\mathcal{M}$ as triples (W, R, V), and frames as pairs (W, R).

Frames you may have seen before

- The "classic" way of thinking about Kripke semantics is to view frames (W, R) as a set W of *possible worlds* linked by a single accessibility relation R.
- In epistemic logic we think of frames $(W, \mathcal{R})$ as a set W of epistemic states linked by a collection of accessibility relations, R^m, one for each agent.
- In tense logic we usually view frames as pairs $(T, <)$. Here we think of T as a set of times or temporal instants, and of $<$ as the earlier-later relation.
- In this course I will also use examples of frames from description logic — frames made from ordinary things.

Satisfaction definition

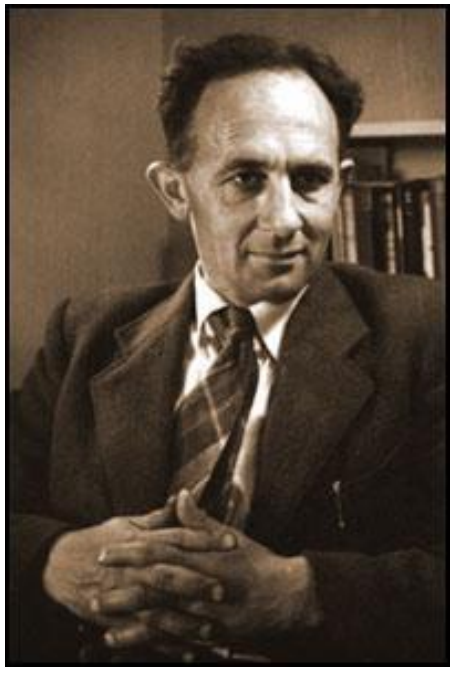
$$
\begin{array}{lll}
\mathcal{M}, w \models \mathrm{p} & \text{iff} & w \in V(\mathrm{p}), \text{ where } \mathrm{p} \in \mathrm{PROP} \\
\mathcal{M}, w \models \neg\varphi & \text{iff} & \mathcal{M}, w \not\models \varphi \\
\mathcal{M}, w \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \varphi \vee \psi & \text{iff} & \mathcal{M}, w \models \varphi \text{ or } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \varphi \rightarrow \psi & \text{iff} & \mathcal{M}, w \not\models \varphi \text{ or } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \langle m \rangle \varphi & \text{iff} & \exists w' (w R^m w' \ \& \ \mathcal{M}, w' \models \varphi) \\
\mathcal{M}, w \models [m] \varphi & \text{iff} & \forall w' (w R^m w' \ \Rightarrow \ \mathcal{M}, w' \models \varphi).
\end{array}
$$

Note the internal perspective: we evaluate formulas inside models, at particular states. Modal formulas are like little creatures (or little automata) that explore models by moving between related points.

Validity

- A formula φ is valid in a model $\mathcal{M} = (W, \mathcal{R}, V)$ iff φ is true at all points in $\mathcal{M}$. Written: $\mathcal{M} \vDash \varphi$.
- A formula φ is valid in a frame $(W, \mathcal{R})$ iff for any model $\mathcal{M} = (W, \mathcal{R})$ based on this frame, $\mathcal{M} \vDash \varphi$. Written: $(W, \mathcal{R}) \vDash \varphi$.
- A formula φ is valid iff for any frame $(W, \mathcal{R})$, $(W, \mathcal{R}) \vDash \varphi$. Written: $\vDash \varphi$.

Tense logic



F means "at some Future time"

G means "it is always Going to be the case"

P means "at some Past time"

H means "it Has always been the case"

$P\text{tini\_busy}$ like *Tini has been busy*

$F\text{tini\_busy}$ like *Tini will be busy*

Description logic

But modal logic is *not* just about abstract objects like possible worlds, epistemic states, and times. It is also used to reason about ordinary things (such as people) and their relationships and properties. This is done in computer science under the name description logic.

This following description logic term

$$\text{student} \sqcap \exists \text{BOSS}.\text{professor}$$

means exactly the same thing as this modal formula

$$\text{student} \wedge \langle \text{BOSS} \rangle \text{professor}.$$

But there's lots of other ways of talking about graphs

- There's nothing magic about frames or Kripke models.
- Frames $(W, \mathcal{R})$ are just a directed multigraphs (or labelled transition systems).
- Valuations simply decorate states with properties.
- So Kripke models for the basic modal language are just (very simple) relational structures in the usual sense of first-order model theory.
- So we don't have to talk about Kripke models using modal logic — we could use first-order logic, or second-order logic, or infinitary logic, or fix-point logic, or indeed any logic interpreted over relational structures.
- Let's see how...

First-order logic for Kripke models

Suppose we have a Kripke model $(W, \mathcal{R}, V)$ for the modal language over MOD and PROP. We talk about this model in first-order logic by making use of the first-order language built from the following symbols:

- For each propositional symbol p, the language contains a unary predicate symbol P. We'll use V to interpret these predicate symbols.
- For each modality $\langle \mathrm{R} \rangle$, the language contains a binary relation symbol R. We'll use the binary relations in $\mathcal{R}$ to interpret these symbols.

The first-order language built over these symbols is called the first-order correspondence language (for the modal language over MOD and PROP).

Doing it first-order style (I)

Consider the modal representation

$$F\mathrm{tini\_busy}.$$

We could use instead the first-order representation

$$\exists t(t_o < t \wedge \mathrm{TINI\_BUSY}(t)).$$

Doing it first-order style (II)

And consider the modal representation

$$\mathsf{student} \wedge \langle \mathrm{BOSS} \rangle \mathsf{professor}.$$

We could instead use the first-order representation

$$\mathsf{STUDENT}(x) \wedge \exists y(\mathrm{BOSS}(x, y) \wedge \mathsf{PROFESSOR}(y)).$$

Standard translation

And in fact, any modal representation can by converted into an equi-satisfiable first-order representation:

$$
\begin{array}{lcl}
\mathrm{ST}_x(p) & = & Px \\
\mathrm{ST}_x(\neg\varphi) & = & \neg\,\mathrm{ST}_x(\varphi) \\
\mathrm{ST}_x(\varphi \wedge \psi) & = & \mathrm{ST}_x(\varphi) \wedge \mathrm{ST}_x(\psi) \\
\mathrm{ST}_x(\varphi \vee \psi) & = & \mathrm{ST}_x(\varphi) \vee \mathrm{ST}_x(\psi) \\
\mathrm{ST}_x(\varphi \rightarrow \psi) & = & \mathrm{ST}_x(\varphi) \rightarrow \mathrm{ST}_x(\psi) \\
\mathrm{ST}_x(\langle R\rangle\varphi) & = & \exists y(Rxy \wedge \mathrm{ST}_y(\varphi)) \\
\mathrm{ST}_x([R]\varphi) & = & \forall y(Rxy \rightarrow \mathrm{ST}_y(\varphi))
\end{array}
$$

Note that $\mathrm{ST}_x(\varphi)$ always contains exactly one free variable, namely x. Also: the y in $\mathrm{ST}_y(\varphi)$ must be new.

Proposition: For any modal formula φ, any Kripke model $\mathcal{M}$, and any state w in $\mathcal{M}$ we have that: $\mathcal{M}, w \vDash \varphi$ iff $\mathcal{M} \vDash \mathrm{ST}_x(\varphi)[x \leftarrow w]$.

Standard translation: Example

$\mathrm{ST}_x(\Diamond\Diamond p \to \Diamond p)$

Standard translation: Example

$$\mathrm{ST}_x(\Diamond\Diamond p \to \Diamond p) \quad = \quad \mathrm{ST}_x(\Diamond\Diamond p) \to \mathrm{ST}_x(\Diamond p)$$

Standard translation: Example

$$\begin{aligned} \mathrm{ST}_x(\Diamond\Diamond p \to \Diamond p) &= \mathrm{ST}_x(\Diamond\Diamond p) \to \mathrm{ST}_x(\Diamond p) \\ &= \exists y(Rxy \wedge \mathrm{ST}_y(\Diamond p)) \to \mathrm{ST}_x(\Diamond p) \end{aligned}$$

Standard translation: Example

$$
\begin{aligned}
\mathrm{ST}_x(\Diamond\Diamond p \to \Diamond p) &= \mathrm{ST}_x(\Diamond\Diamond p) \to \mathrm{ST}_x(\Diamond p) \\
&= \exists y(Rxy \wedge \mathrm{ST}_y(\Diamond p)) \to \mathrm{ST}_x(\Diamond p) \\
&= \exists y(Rxy \wedge \mathrm{ST}_y(\Diamond p)) \to \exists z(Rxz \wedge \mathrm{ST}_z(p))
\end{aligned}
$$

Standard translation: Example

$$
\begin{aligned}
\mathrm{ST}_x(\Diamond\Diamond p \to \Diamond p) &= \mathrm{ST}_x(\Diamond\Diamond p) \to \mathrm{ST}_x(\Diamond p) \\
&= \exists y(Rxy \wedge \mathrm{ST}_y(\Diamond p)) \to \mathrm{ST}_x(\Diamond p) \\
&= \exists y(Rxy \wedge \mathrm{ST}_y(\Diamond p)) \to \exists z(Rxz \wedge \mathrm{ST}_z(p)) \\
&= \exists y(Rxy \wedge \mathrm{ST}_y(\Diamond p)) \to \exists z(Rxz \wedge Pz)
\end{aligned}
$$

Standard translation: Example

$$
\begin{aligned}
\mathrm{ST}_x(\Diamond\Diamond p \to \Diamond p) &= \mathrm{ST}_x(\Diamond\Diamond p) \to \mathrm{ST}_x(\Diamond p) \\
&= \exists y(Rxy \wedge \mathrm{ST}_y(\Diamond p)) \to \mathrm{ST}_x(\Diamond p) \\
&= \exists y(Rxy \wedge \mathrm{ST}_y(\Diamond p)) \to \exists z(Rxz \wedge \mathrm{ST}_z(p)) \\
&= \exists y(Rxy \wedge \mathrm{ST}_y(\Diamond p)) \to \exists z(Rxz \wedge Pz) \\
&= \exists y(Rxy \wedge \exists v(Ryv \wedge \mathrm{ST}_v(p)) \to \exists z(Rxz \wedge Pz)
\end{aligned}
$$

Standard translation: Example

$$
\begin{aligned}
\mathrm{ST}_x(\Diamond\Diamond p \to \Diamond p) &= \mathrm{ST}_x(\Diamond\Diamond p) \to \mathrm{ST}_x(\Diamond p) \\
&= \exists y(Rxy \wedge \mathrm{ST}_y(\Diamond p)) \to \mathrm{ST}_x(\Diamond p) \\
&= \exists y(Rxy \wedge \mathrm{ST}_y(\Diamond p)) \to \exists z(Rxz \wedge \mathrm{ST}_z(p)) \\
&= \exists y(Rxy \wedge \mathrm{ST}_y(\Diamond p)) \to \exists z(Rxz \wedge Pz) \\
&= \exists y(Rxy \wedge \exists v(Ryv \wedge \mathrm{ST}_v(p)) \to \exists z(Rxz \wedge Pz) \\
&= \exists y(Rxy \wedge \exists v(Ryv \wedge Pv)) \to \exists z(Rxz \wedge Pz)
\end{aligned}
$$

As promised, $\mathrm{ST}_x(\varphi)$ contains exactly one free variable (namely x).

Standard translation: Example

$$
\begin{aligned}
\mathrm{ST}_x(\Diamond\Diamond p \to \Diamond p) &= \mathrm{ST}_x(\Diamond\Diamond p) \to \mathrm{ST}_x(\Diamond p) \\
&= \exists y(Rxy \wedge \mathrm{ST}_y(\Diamond p)) \to \mathrm{ST}_x(\Diamond p) \\
&= \exists y(Rxy \wedge \mathrm{ST}_y(\Diamond p)) \to \exists z(Rxz \wedge \mathrm{ST}_z(p)) \\
&= \exists y(Rxy \wedge \mathrm{ST}_y(\Diamond p)) \to \exists z(Rxz \wedge Pz) \\
&= \exists y(Rxy \wedge \exists v(Ryv \wedge \mathrm{ST}_v(p)) \to \exists z(Rxz \wedge Pz) \\
&= \exists y(Rxy \wedge \exists v(Ryv \wedge Pv)) \to \exists z(Rxz \wedge Pz)
\end{aligned}
$$

As promised, $\mathrm{ST}_x(\varphi)$ contains exactly one free variable (namely x).

Also note: only two variables, x and y, are really needed:

$$\exists y(Rxy \wedge \exists v(Ryv \wedge Pv)) \to \exists z(Rxz \wedge Pz)$$

$$\exists y(Rxy \wedge \exists x(Ryx \wedge Px)) \to \exists y(Rxy \wedge Py)$$

So the Standard Translation shows that modal logic actually translates into F0-2, the two-variable fragment of first-order logic.

So aren't we better off with first-order logic . . . ?

- We've just seen that any modal formula can be systematically converted into an equi-satisfiable first-order formula.
- The reverse is not true: first-order logic can describe models in far more detail that modal logic can. (This is already clear from our observation that the standard translation takes us to F0-2.) When it comes to talking about models, some first-order formulas have no modal equivalent. Modal languages are weaker than their corresponding first-order languages.
- So why bother with modal logic?

Reasons for going modal

- **Computability**. First-order logic is undecidable over arbitrary models. Modal logic is decidable over arbitrary models (indeed, decidable in PSPACE). Modal logic trades expressivity for computability.
- **Simplicity**. The standard translation shows us that modalities are essentially 'macros' encoding a quantification over related states. Modal notation hides the bound variables, resulting in compact, easy to read, rules and representations.
- **Internal perspective**. A natural way of thinking about models. And taken seriously, leads to an elegant characterization of what modal logic can say about models.

Let's take a closer look at simplicity and the internal perspective.

Simplicity shows in modal axiomatics

Here's the **K**-axiom system (for a basic modal language with a single $\Box$ and $\Diamond$ pair): simply add the following axioms and rules (the **normality** axioms/rules) to propositional logic:

Axioms: $\Box(p \to q) \to (\Box p \to \Box q)$

Rules:

MP: If $\vdash \varphi \to \psi$ and $\vdash \varphi$ then $\vdash \psi$

$\text{Gen}_{\Box}$: If $\vdash \varphi$ then $\vdash \Box\varphi$

Sub_{prop}: Uniform substitution of formulas for propositional variables

Soundness and completeness:
φ is **K**-provable iff φ is true at every world in every model.

Simple axiom systems for many other logics

In applications we may be interested in formulas true in all transitive models, or all reflexive models, or all symmetric models, or . . . lots of other interesting models. We can often capture these logics simply by adding appropriate axioms to **K**. Here's a selection:

T	$p \to \Diamond p$	Reflexivity
B	$p \to \Box\Diamond p$	Symmetry
4	$\Diamond\Diamond p \to \Diamond p$	Transitivity

Soundness and completeness:

φ is **T**-provable iff φ is true at every world in every reflexive model.

φ is **B**-provable iff φ is true at every world in every symmetric model.

φ is **K4**-provable iff φ is true at every world in every transitive model.

When all goes well, proving completeness is easy: as these formulas impose their meaning on the canonical model. This brings us to an important concept. . .

Frame definability

We say that a formula φ defines a class of frames F if:

$$(W, R) \vDash \varphi \quad \text{iff} \quad (W, R) \text{ belongs to F.}$$

T	$p \to \Diamond p$	defines the class of reflexive frames
B	$p \to \Box\Diamond p$	defines the class of symmetric frame
4	$\Diamond\Diamond p \to \Diamond p$	defines the class of transitive frames

That is: $p \to \Diamond p$ defines the class of reflexive frames (that is, defines reflexivity) because:

- $p \to \Diamond p$ is valid on any reflexive frame.
- If a frame is *not* reflexive — that is, it contains some state w such that not wRw — then we can find a valuation that falsifies $p \to \Diamond p$ at some state.

Internal perspective: Bisimulation (I)

Fundamental notion of equivalence between states for modal logic

Bisimulations are used in other areas than modal logic. Their role in them all is to provide an appropriate notion of equivalence.

Social Network Theory Bisimulations are the "regular equivalences" of social network theory, which partition networks into sets of actors with the same social role. Marx and Masuch, *Regular equivalence and dynamic logic*, Social Networks, 25(1):51–65, Elsevier, 2003.

Non-well-founded Set Theory Here bisimulations replace extensionality, the criterion of standard set equality: two *non-well-founded* sets (graphs) are equal iff they are bisimilar. Look for work by: Peter Aczel, Larry Moss, Jon Barwise.

Theoretical Computer Science A huge literature here. Basically, bisimulations model the notion of behavioural equivalence for processes. Hennessy and Milner, *Algebraic laws for nondeterminism and concurrency*, Journal of the ACM (JACM), 01 January 1985, Vol.32(1), pp.137-161 is particularly relevant to modal logic.

Internal perspective: Bisimulation (II)

Let $\mathcal{M} = (W, R, V)$ and $\mathcal{M}' = (W', R', V')$ be models for the basic modal language (with just one $\Box$ and $\Diamond$). A relation $Z \subseteq W \times W'$ is a bisimulation between $\mathcal{M}$ and $\mathcal{M}'$ if the following conditions are met:

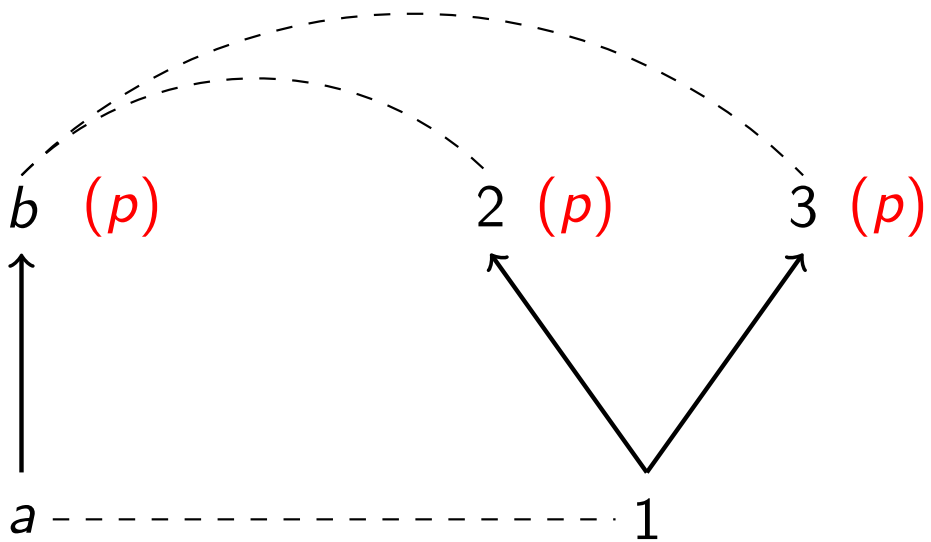
1. Atomic Harmony: if wZw' then $w \in V(p)$ iff $w' \in V'(p)$, for all propositional symbols p.
2. Forth: if wZw' and wRv then there is a v' such that $w'R'v'$ and vZv'.
3. Back: if wZw' and $w'R'v'$ then there is a v such that wRv and vZv'.

Internal perspective: Bisimulation example (I)

Model 1: $W = \{a, b\}$, $R = \{(a, b)\}$ and $V(p) = \{b\}$, for all proposition letters p.

Model 2: $W = \{1, 2, 3\}$, $R = \{(1, 2), (1, 3)\}$ and $V(p) = \{2, 3\}$, for all proposition letters p.

Define Z to be: $\{(a, 1), (b, 2), (b, 3)\}$. Then Z is a bisimulation.

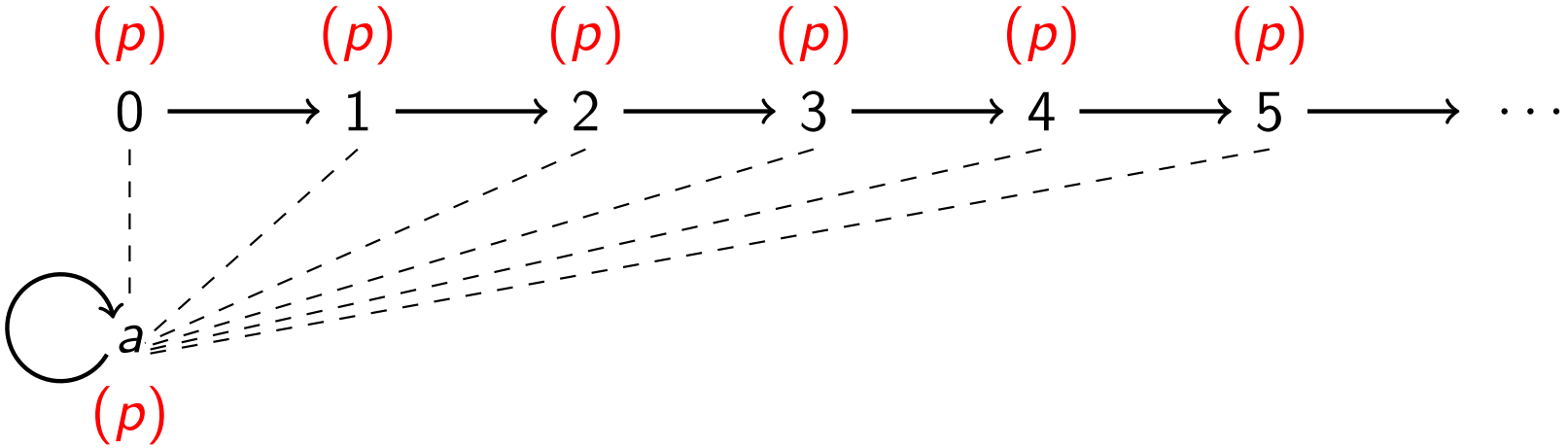


Internal perspective: Bisimulation example (II)

Model 1: The natural numbers under their usual ordering, with all proposition letters p true everywhere.

Model 2: $W = \{a\}$, $R = \{(a, a)\}$ and $V(p) = \{a\}$, for all proposition letters p.

Define Z to be: $\{(0, a), (1, a), (2, a), (3, a), \ldots\}$. Then Z is a bisimulation.



Modal formulas are invariant under bisimulation

Proposition: Let $\mathcal{M} = (W, R, V)$ and $\mathcal{M}' = (W', R', V')$ be models for the basic modal language (with just one $\Box$ and $\Diamond$), and let Z be a bisimulation between $\mathcal{M}$ and $\mathcal{M}'$. Then for all modal formulas φ, and all points w in $\mathcal{M}$ and w' in $\mathcal{M}'$ such that w is bisimilar to w':

$$\mathcal{M}, w \models \varphi \text{ iff } \mathcal{M}', w' \models \varphi.$$

In words: bisimilar points are modally equivalent, or to put it another way: modal formulas are invariant under bisimulations.

Proof: Induction on the structure of φ.

Not all first-order formulas are bisimulation invariant

- A first-order formula in one free variable $\varphi(x)$ is bisimulation-invariant if for all bisimulations Z between models $\mathcal{M}$ and $\mathcal{M}'$, if wZw' then $\mathcal{M} \models \varphi[w]$ iff $\mathcal{M}' \models \varphi[w']$.
- Not all first-order formulas are bisimulation invariant (so once again: we see that not all first-order formulas can be translated into modal formulas).
- For example: Rxx is not invariant under bisimulation (as we have just seen).
- But bisimulation invariance seems to be useful in various domains, so it is natural to ask: precisely which first-order formulas are bisimulation invariant? The answer is elegant ...

The van Benthem Characterization Theorem

For all first-order formulas φ (in the correspondence language) containing exactly one free variable, φ is bisimulation-invariant iff φ is equivalent to the standard translation of a modal formula.
In short, modal logic is a simple notation for capturing exactly the bisimulation-invariant fragment of first-order logic.

Proof:
($\Rightarrow$) Immediate from the invariance of modal formula under bisimulation.
($\Leftarrow$) Non-trivial (usually proved using elementary chains or by appealing to the existence of saturated models).

Many proofs available. The original proof is particularly elegant: Theorem 3.9, pages 43–46, *Modal Logic and Classical Logic*, Johan van Benthem, Bibliopolis, ISBN 88-7088-113-X.

Back to slogan 3

Slogan 3: Modal languages are not isolated formal systems.

Modal languages over models are essentially simple fragments of first-order logic. These fragments have a number of attractive properties such as robust decidability and bisimulation invariance. Traditional modal notation is essentially a nice (quantifier free) 'macro' notation for working with this fragment.

Back to slogan 2

Slogan 2: Modal languages provide an internal, local perspective on relational structures.

This is not just an intuition: the notion of bisimulation, and the results associated with it show that this is the key model theoretic fact at work in modal logic.

Back to slogan 1

Slogan 1: Modal languages are simple yet expressive languages for talking about relational structures.

You can use modal logic for just about anything. Anywhere you see a graph, you can use a modal language to talk about it.

That was the good news — now comes the bad

Orthodox modal languages have an obvious drawback for many applications: they don't let us refer to individual states (worlds, times, situations, nodes, ...). That is, they don't allow us to say things like

- this happened *there*; or
- this happened *then*; or
- *this* state has property φ; or
- node *peter's-printer* is marked *out-of-paper*.

and so on.

Temporal logic

- In computer science applications, we often need to reason about the time when something happens. Representations of the form **Holds**(e, t) are common. Ordinary Priorean tense logic does not let us do this.
- Moreover, *Tini sighed and started writing again* doesn't just mean $P\mathrm{tini\_sigh\_start\_write\_again}$. Part of its meaning comes from the context of utterance.

Tense in text

Tini paused. She was so tired. Tini sighed and started to write again.

The states described by the first two sentences hold at the same time. The event described by the second takes place a little later. In orthodox modal logics there is no way to assert the identity of the times needed for the first two sentences, nor to capture the move forward in time needed by the third.

For this reason modal languages for temporal representation have not been the tool of choice in natural language semantics for over 30 years.

Description logic (I)

As we have already said, there is a transparent correspondence between simple description logic terms and modal formulas:

$$\text{student} \sqcap \exists \text{BOSS}.\text{professor}$$

$$\text{student} \wedge \langle \text{BOSS} \rangle \text{professor}$$

Nonetheless, this correspondence only involves what description logicians call the TBox (Terminological Box), where general concepts are defined.

Description logic (II)

Orthodox modal logic does not have anything to say about the ABox (Assertional Box):

$$\text{tini} : \text{Clever}$$

$$(\text{andi}, \text{dimas}) : \text{Friends}$$

That is, it can't make assertions about individuals, for it has no tools for naming individuals.

Ambition

- We want to be able to refer to states, but want to do so without destroying the simplicity of propositional modal logic.
- But how can we do this — propositional modal logic has very few moving parts?
- Answer: sort the atomic symbols. Use formulas as terms.
- This solution can be traced back to work in the 1960s by Arthur Prior (lots more on Prior's work is coming up in Lecture 5).
- This solution will fix the obvious shortcoming — and, as we shall see in the next lecture — will fix a lot more besides.

Extension #1

- Take a language of basic modal logic (with propositional variables p, q, r, and so on) and add a second sort of atomic formula.
- The new atoms are called nominals, and are typically written i, j, k, and l.
- Both types of atom can be freely combined to form more complex formulas in the usual way; for example,

$$\Diamond(i \wedge \mathrm{p}) \wedge \Diamond(i \wedge \mathrm{q}) \rightarrow \Diamond(\mathrm{p} \wedge \mathrm{q})$$

  is a well formed formula.
- Insist that each nominal be true at exactly one world in any model. A nominal names a state by being true there and nowhere else.

We already have a richer logic

Consider the orthodox formula

$$\Diamond(\mathrm{r} \wedge \mathrm{p}) \wedge \Diamond(\mathrm{r} \wedge \mathrm{q}) \rightarrow \Diamond(\mathrm{p} \wedge \mathrm{q})$$

This is easy to falsify.

We already have a richer logic

Consider the orthodox formula

$$\Diamond(\mathrm{r} \wedge \mathrm{p}) \wedge \Diamond(\mathrm{r} \wedge \mathrm{q}) \rightarrow \Diamond(\mathrm{p} \wedge \mathrm{q})$$

This is easy to falsify.

On the other hand, the hybrid formula

$$\Diamond(i \wedge \mathrm{p}) \wedge \Diamond(i \wedge \mathrm{q}) \rightarrow \Diamond(\mathrm{p} \wedge \mathrm{q})$$

is valid (unfalsifiable in any model). Nominals name, and this adds to the expressive power at our disposal.

Extension #2

- Add formulas of form $@_i\varphi$.
- Such formulas assert that φ is satisfied at the point named by the nominal i.
- Expressions of the form $@_i$ are usualll called satisfaction operators or at operators.

Let's make these ideas precise ...

Hybrid logic: Syntax

- Given ordinary propositional symbols PROP $= \{\mathrm{p}, \mathrm{q}, \mathrm{r}, \ldots\}$, and modalities MOD, let NOM $= \{i, j, k, l, \ldots\}$ be a non-empty set disjoint from PROP.
- The elements of NOM are called nominals; they are a second sort of atomic symbol which will be used to name states. The basic hybrid language (over PROP, MOD, and NOM) is defined as follows:

$$\begin{aligned} \mathrm{WFF} \quad := \quad & i \mid p \mid \top \mid \bot \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \\ & \mid \varphi \rightarrow \psi \mid \langle \mathrm{M} \rangle \varphi \mid [\mathrm{M}] \varphi \mid @_i \varphi \end{aligned}$$

Hybrid logic: Semantics

- As before, a model $\mathcal{M}$ is a triple $(W, \mathcal{R}, V)$.
- As before, $(W, \mathcal{R})$ is just a frame (a labelled transition system).
- The difference lies in V. Now a valuation V is a function with domain PROP$\cup$NOM and range $Pow(W)$ such that for all $i \in$ NOM, $V(i)$ is a singleton subset of W.
- That is, a valuation makes each nominal true at a unique state; the nominal labels this state by being true there and nowhere else.
- We call the unique state w that belongs to $V(i)$ the denotation of i under V.

Satisfaction definition

$\mathcal{M}, w \models a$	iff	$w \in V(a)$, where $a \in \mathrm{PROP} \cup \mathrm{NOM}$
$\mathcal{M}, w \models \neg\varphi$	iff	$\mathcal{M}, w \not\models \varphi$
$\mathcal{M}, w \models \varphi \wedge \psi$	iff	$\mathcal{M}, w \models \varphi$ and $\mathcal{M}, w \models \psi$
$\mathcal{M}, w \models \varphi \vee \psi$	iff	$\mathcal{M}, w \models \varphi$ or $\mathcal{M}, w \models \psi$
$\mathcal{M}, w \models \varphi \rightarrow \psi$	iff	$\mathcal{M}, w \not\models \varphi$ or $\mathcal{M}, w \models \psi$
$\mathcal{M}, w \models \langle \mathrm{M} \rangle \varphi$	iff	$\exists w'(wR^m w' \ \& \ \mathcal{M}, w' \models \varphi)$
$\mathcal{M}, w \models [\mathrm{M}] \varphi$	iff	$\forall w'(wR^m w' \ \Rightarrow \ \mathcal{M}, w' \models \varphi)$.
$\mathcal{M}, w \models @_i\varphi$	iff	$\mathcal{M}, \mathrm{i} \models \varphi$, where i is the denotation of i under V.

Temporal logic

- Looks closer to AI temporal representation formalisms: @ can play the role of **Holds**.
- And we can now handle natural language examples more convincingly: $P(i \wedge \text{tini\_sigh\_start\_write\_again})$ locates the trigger-squeezing not merely in the past, but at a specific temporal state there, namely the one named by i, which lets us represent more of the meaning of *Tini sighed and started to write again.*. Let's take this a little further...

Tense in text

Tini paused. She was so tired. Tini sighed and started to write again.

Tense in text

Tini paused. She was so tired. Tini sighed and started to write again.

$$P(i \wedge \text{tini-pause})$$

Tense in text

Tini paused. She was so tired. Tini sighed and started to write again.

$$P(i \wedge \text{tini-pause})$$

$$\wedge \quad P(j \wedge \text{she-tired})$$

Tense in text

Tini paused. She was so tired. Tini sighed and started to write again.

$$P(i \wedge \text{tini-pause})$$

$$\wedge \quad P(j \wedge \text{she-tired}) \qquad \wedge \quad @_j i$$

Tense in text

Tini paused. She was so tired. Tini sighed and started to write again.

$$
\begin{array}{lll}
 & P(i \wedge \text{tini-pause}) & \\
\wedge & P(j \wedge \text{she-tired}) & \wedge \quad @_j i \\
\wedge & P(k \wedge \text{tini-sigh-start-write-again}) &
\end{array}
$$

Tense in text

Tini paused. She was so tired. Tini sighed and started to write again.

$$
\begin{array}{lll}
 & P(i \wedge \text{tini-pause}) & \\
\wedge & P(j \wedge \text{she-tired}) & \wedge \quad @_j i \\
\wedge & P(k \wedge \text{tini-sigh-start-write-again}) & \wedge \quad @_k P i
\end{array}
$$

Description logic (I)

We can now make ABox statements. For example, to capture the effect of the (conceptual) ABox assertion

$$\text{tini} : \text{clever}$$

we can write

$$@_{\text{tini}}\text{clever}$$

Description logic (II)

Similarly, to capture the effect of the (relational) ABox assertion

$$(\mathsf{andi}, \mathsf{dimas}) : \mathsf{Friends}$$

we can write

$$@_{\mathsf{andi}} \langle \mathsf{Friends} \rangle \mathsf{dimas}$$

Basic hybrid language clearly modal

Neither syntactical nor computational simplicity, nor general 'style' of modal logic, has been compromised.

- Nominals are atomic formulas.
- Satisfaction operators follow all the normality rules/axioms that we saw for $\Box$. That is, for any nominal i we have that:
  - $@_i(\varphi \to \psi) \to (@_i\varphi \to @_i\psi)$ is valid.
  - If φ is valid, then so is $@_i\varphi$.
- Indeed, satisfaction operators are even self-dual modal operators: $@_i\phi$ and $\neg@_i\neg\phi$ say exactly the same thing.

Basic hybrid logic is computable

Enriching ordinary propositional modal logic with both nominals and satisfaction operators does not affect computability. The basic hybrid logic is decidable. Indeed we even have:

Theorem: The satisfiability problem for basic hybrid languages over arbitrary models is PSPACE-complete.[1]

That is (up to a polynomial) the hybridized language has the same complexity as the orthodox modal language we started with.

[1] Theorem 1 in "A roadmap of the complexity of hybrid logics", Carlos Areces, Patrick Blackburn, and Maarten Marx, Computer science logic (J. Flum and M. Rodriguez-Artalejo, editors), LNCS, no. 1683, Springer, 1999, Proceedings of the 8th Annual Conference of the EACSL, Madrid, September 1999, pp. 307-321.

Standard translation

Any basic hybrid formula can by converted into an equi-satisfiable first-order formula. All we have to do is add a first-order constant (or variable) i for each nominal i and translate as follows (note the use of equality):

$$
\begin{array}{lcl}
\mathrm{ST}_x(p) & = & Px \\
\mathrm{ST}_x(i) & = & (i = x) \\
\mathrm{ST}_x(\neg\varphi) & = & \neg\,\mathrm{ST}_x(\varphi) \\
\mathrm{ST}_x(\varphi \wedge \psi) & = & \mathrm{ST}_x(\varphi) \wedge \mathrm{ST}_x(\psi) \\
\mathrm{ST}_x(\varphi \vee \psi) & = & \mathrm{ST}_x(\varphi) \vee \mathrm{ST}_x(\psi) \\
\mathrm{ST}_x(\varphi \rightarrow \psi) & = & \mathrm{ST}_x(\varphi) \rightarrow \mathrm{ST}_x(\psi) \\
\mathrm{ST}_x(\langle R\rangle\varphi) & = & \exists y(Rxy \wedge \mathrm{ST}_y(\varphi)) \\
\mathrm{ST}_x([R]\varphi) & = & \forall y(Rxy \rightarrow \mathrm{ST}_y(\varphi)) \\
\mathrm{ST}_x(@_i\varphi) & = & \mathrm{ST}_i(\varphi)
\end{array}
$$

Note that if we translate nominals by constants, $\mathrm{ST}_x(\varphi)$ always contains at most one free variable (namely x).

Proposition: For any basic hybrid formula φ, any Kripke model $\mathcal{M}$, and any state w in $\mathcal{M}$, we have that: $\mathcal{M}, w \vDash \varphi$ iff $\mathcal{M} \vDash \mathrm{ST}_x(\varphi)[x \leftarrow w]$.

Standard translation: Example

$\mathrm{ST}_x((@_i p \wedge i) \rightarrow p)$

Standard translation: Example

$$\mathrm{ST}_x((@_i p \wedge i) \rightarrow p) \quad = \quad (\mathrm{ST}_x(@_i p \wedge i) \rightarrow \mathrm{ST}_x(p))$$

Standard translation: Example

$$
\begin{aligned}
\mathrm{ST}_x((@_i p \wedge i) \rightarrow p) &= (\mathrm{ST}_x(@_i p \wedge i) \rightarrow \mathrm{ST}_x(p)) \\
&= (\mathrm{ST}_x(@_i p) \wedge \mathrm{ST}_x(i)) \rightarrow \mathrm{ST}_x(p)
\end{aligned}
$$

Standard translation: Example

$$
\begin{aligned}
\mathrm{ST}_x((@_i p \wedge i) \rightarrow p) &= (\mathrm{ST}_x(@_i p \wedge i) \rightarrow \mathrm{ST}_x(p)) \\
&= (\mathrm{ST}_x(@_i p) \wedge \mathrm{ST}_x(i)) \rightarrow \mathrm{ST}_x(p) \\
&= (\mathrm{ST}_x(@_i p) \wedge \mathrm{ST}_x(i)) \rightarrow Px
\end{aligned}
$$

Standard translation: Example

$$
\begin{aligned}
\mathrm{ST}_x((@_i p \wedge i) \to p) &= (\mathrm{ST}_x(@_i p \wedge i) \to \mathrm{ST}_x(p)) \\
&= (\mathrm{ST}_x(@_i p) \wedge \mathrm{ST}_x(i)) \to \mathrm{ST}_x(p) \\
&= (\mathrm{ST}_x(@_i p) \wedge \mathrm{ST}_x(i)) \to Px \\
&= (\mathrm{ST}_x(@_i p) \wedge i = x) \to Px
\end{aligned}
$$

Standard translation: Example

$$
\begin{aligned}
\mathrm{ST}_x((@_i p \wedge i) \to p) &= (\mathrm{ST}_x(@_i p \wedge i) \to \mathrm{ST}_x(p)) \\
&= (\mathrm{ST}_x(@_i p) \wedge \mathrm{ST}_x(i)) \to \mathrm{ST}_x(p) \\
&= (\mathrm{ST}_x(@_i p) \wedge \mathrm{ST}_x(i)) \to Px \\
&= (\mathrm{ST}_x(@_i p) \wedge i = x) \to Px \\
&= (\mathrm{ST}_i(p) \wedge i = x) \to Px
\end{aligned}
$$

Standard translation: Example

$$
\begin{aligned}
\mathrm{ST}_x((@_i p \wedge i) \rightarrow p) &= (\mathrm{ST}_x(@_i p \wedge i) \rightarrow \mathrm{ST}_x(p)) \\
&= (\mathrm{ST}_x(@_i p) \wedge \mathrm{ST}_x(i)) \rightarrow \mathrm{ST}_x(p) \\
&= (\mathrm{ST}_x(@_i p) \wedge \mathrm{ST}_x(i)) \rightarrow Px \\
&= (\mathrm{ST}_x(@_i p) \wedge i = x) \rightarrow Px \\
&= (\mathrm{ST}_i(p) \wedge i = x) \rightarrow Px \\
&= (Pi \wedge i = x) \rightarrow Px
\end{aligned}
$$

It should be clear form this example that nominals behave very much like first-order constants (or free variables). It should also be becoming clear that basic hybrid logic essentially offers us a tool for "adding naming and equality reasoning to modal logic", or a "modal logic of equality — for the elements in Kripke models".

Standard translation: An alternative

In Lecture 4 we will meet the downarrow binder $\downarrow$, and when we do, it will be instructive to compare its standard translation with the following variant translation for @:

$$\mathrm{ST}_x(@_i\varphi) \quad = \quad \exists y(y = i \wedge \mathrm{ST}_y(\varphi)).$$

Clearly this yields equivalent (though slightly longer) translations. For example, on the previous slide we saw that:

$$\mathrm{ST}_x((@_i p \wedge i) \rightarrow p) \quad = \quad (Pi \wedge i = x) \rightarrow Px$$

Under the alternative translation we would obtain:

$$\mathrm{ST}_x((@_i p \wedge i) \rightarrow p) \quad = \quad (\exists y(y = i \wedge Py) \wedge i = x) \rightarrow Px$$

But until then, we will continue to work with the simpler translation on the previous slide.

Basic hybrid logic can specify Robinson Diagrams

- $@_i p$ says that the states labelled i bears the information p, while $\neg @_i p$ denies this. That is, they are modal specifications of how atomic properties are distributed.
- $@_i j$ says that the states labelled i and j are identical, while $\neg @_i j$ says they are distinct. That is, they are modal specifications of theories of state equality.
- $@_i \Diamond j$ says that the state labelled j is a successor of the state labelled i, and $\neg @_i \Diamond j$ denies this. That is, they are modal specifications of theories of state succession.

That is, we have all the tools needed to completely describe models (that is, what classical model theorists call Robinson diagrams). This makes life straightforward when it comes to proving completeness (and, with downarrow, interpolation) results.

An important concept: pure formulas

- A formula of the basic hybrid language is pure if it contains no propositional variables. That is, the only atoms in pure formulas are nominals (and $\top$ and $\bot$ if we have them in the language).
- Pure formulas are interesting expressively — as we shall now see.
- But they are also useful for inference, as we shall see tomorrow and the day after.

Frame definability (I)

Recall that a formula defines a class of frames if it is valid on precisely the frames belonging to that class. We can define many important classes of frames using pure formulas:

$@_i \Diamond i$	*Reflexivity*
$@_i \Diamond j \rightarrow @_j \Diamond i$	*Symmetry*
$@_i \Diamond j \wedge @_j \Diamond k \rightarrow @_i \Diamond k$	*Transitivity*

Note: other pure formulas can be used. For example, $\Diamond\Diamond i \rightarrow \Diamond i$, also defines transitivity, and $i \rightarrow \Diamond i$ also defines reflexivity.

Frame definability (II)

These previous three examples are also definable using orthodox modal language. But pure formulas can also define frame classes which are not definable in orthodox modal logic:

$@_i \neg \Diamond i$	*Irreflexivity*
$@_i \Diamond j \rightarrow @_j \neg \Diamond i$	*Asymmetry*
$@_i \Box (\Diamond i \rightarrow i)$	*Antisymmetry*
$@_j \Diamond i \vee @_j i \vee @_i \Diamond j$	*Trichotomy*

Note: other pure formulas can be used. For example, $i \rightarrow \neg \Diamond i$ also defines irreflexivity.

But what *is* basic hybrid logic?

We have seen many examples of what basic hybrid logic can do in various applications.

We've also seen that a number of the properties we liked about modal logic are inherited by the basic hybrid language.

This is all very nice — but none of it gives us a clear mathematical characterization of what basic hybrid logic actually is.

And it is possible to give such a characterization, and a genuinely modal one at that. Let's take a look ...

Bisimulation-with-constants

Let $\mathcal{M} = (W, R, V)$ and $\mathcal{M}' = (W', R', V')$ be models for the basic modal language (with just one $\Box$ and $\Diamond$). A relation $Z \subseteq W \times W'$ is a bisimulation-with-constants between $\mathcal{M}$ and $\mathcal{M}'$ if the following conditions are met:

1. Atomic Harmony: if wZw' then $w \in V(p)$ iff $w' \in V'(p)$, for all propositional symbols p, and all nominals i.
2. Forth: if wZw' and wRv then there is a v' such that $w'R'v'$ and vZv'.
3. Back: if wZw' and $w'R'v'$ then there is a v such that wRv and vZv'.
4. Nominal Constancy: Any two points named by the same nominal are related by Z.

Basic hybrid formulas are invariant under bisimulations-with-constants

Proposition: Let $\mathcal{M} = (W, R, V)$ and $\mathcal{M}' = (W', R', V')$ be models for the basic hybrid language (with just one $\Box$ and $\Diamond$), and let Z be a bisimulation-with-constants between $\mathcal{M}$ and $\mathcal{M}'$. Then for all basic hybrid formulas φ, and all points w in $\mathcal{M}$ and w' in $\mathcal{M}'$ such that w is bisimilar to w':

$$\mathcal{M}, w \models \varphi \text{ iff } \mathcal{M}', w' \models \varphi.$$

Proof: Induction on the structure of φ.

Lifting the van Benthem Characterization Theorem

For all first-order formulas φ (in the correspondence language with constants and equality) containing at most one free variable, φ is bisimulation-with-constants invariant iff φ is equivalent to the standard translation of a basic hybrid formula.[2]

In short, basic hybrid logic is a simple notation for capturing exactly the bisimulation-invariant fragment of first-order logic when we make use of constants and equality.

Proof:
($\Rightarrow$) Immediate from the invariance of hybrid formulas under bisimulation.
($\Leftarrow$) Can be proved using elementary chains or by appealing to the existence of saturated models.

---

[2] See "Hybrid logics: Characterization, interpolation and complexity", Carlos Areces, Patrick Blackburn, and Maarten Marx, *The Journal of Symbolic Logic*, 66(3), pp. 977-1010, 2001.

Summing up . . .

- We learned about some of the good points of orthodox modal logic, but also saw that its inability to refer to states is a weakness for various applications.
- We saw that adding nominals and satisfaction operators fixes these weaknesses without sacrificing what we liked about modal logic in the first place. Basic hybrid logic is a natural generalization of orthodox modal logic.
- But as we shall soon learn, hybridization has fixed some less obvious shortcomings of orthodox modal logic too. In particular, it has given us a logical formalism that is easy to use deductively — as we shall see tomorrow.