Introduction to Hybrid Logic

Atelier Jeunes Chercheurs
Semaine Nancéienne de Sémantique Formelle
LORIA/INRIA Nancy-Grand Est
22 March 2010

Patrick Blackburn
patrick.blackburn@loria.fr

Goals of the course

This mini-course (or, more accurately, extended lecture) introduces and explores hybrid logic, a form of modal logic in which it is possible to name worlds (or times, or computational states, or situations, or nodes in parse trees, or people — indeed, whatever it is that the elements of Kripke Models are taken to represent).

The course has two main goals. The first is to convey, as clearly as possible, the ideas and intuitions that have guided the development of hybrid logic. The second is to gently hint at some technical themes, such as the role of bisimulation and why hybrid logic is so useful proof theoretically.

All that — and in only three hours too...!

To give a little more detail...

In today's lecture we discuss:

- Orthodox modal logic — from an Amsterdam perspective.
- A problem with orthodox modal logic.
- Fixing this problem with basic hybrid logic.
- Why basic hybrid logic is genuinely modal: bisimulations.
- Why basic hybrid logic is good for your proof theory: tableau systems.
- Flagging the here and now: the downarrow binder

Go to `http://webloria.loria.fr/~blackbur/jsm.pdf` for the slides; there's more in the slides than I am likely to cover in the lecture.

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.

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:

$$\begin{array}{rcl} \text{WFF} & := & \mathrm{p} \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \\ & & \mid \varphi \rightarrow \psi \mid \langle m \rangle \varphi \mid [m]\varphi \end{array}$$

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

$[m]\varphi$ can be regarded as shorthand for $\neg\langle m \rangle \neg\varphi$. Sometimes useful to add primitive atomic symbols $\top$ (true) and $\bot$ (false).

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 possible worlds, or epistemic states, or times, or states in a transition system, or geometrical points, or people standing in various relationships, or nodes in a parse tree — indeed, pretty much anything you like.
  - $\mathcal{R}$ is a collection of binary relation on W (one for each modality)
  - V is a valuation assigning subsets of W to propositional symbols.
- The component $(W, \mathcal{R})$ traditionally call a frame.

Satisfaction Definition

$$\begin{array}{lll} \mathcal{M}, w \Vdash \mathrm{p} & \text{iff} & w \in V(\mathrm{p}), \text{ where } \mathrm{p} \in \text{PROP} \\ \mathcal{M}, w \Vdash \neg\varphi & \text{iff} & \mathcal{M}, w \nVdash \varphi \\ \mathcal{M}, w \Vdash \varphi \wedge \psi & \text{iff} & \mathcal{M}, w \Vdash \varphi \text{ and } \mathcal{M}, w \Vdash \psi \\ \mathcal{M}, w \Vdash \varphi \vee \psi & \text{iff} & \mathcal{M}, w \Vdash \varphi \text{ or } \mathcal{M}, w \Vdash \psi \\ \mathcal{M}, w \Vdash \varphi \rightarrow \psi & \text{iff} & \mathcal{M}, w \nVdash \varphi \text{ or } \mathcal{M}, w \Vdash \psi \\ \mathcal{M}, w \Vdash \langle m \rangle \varphi & \text{iff} & \exists w' (w R^m w' \;\&\; \mathcal{M}, w' \Vdash \varphi) \\ \mathcal{M}, w \Vdash [m]\varphi & \text{iff} & \forall w' (w R^m w' \;\Rightarrow\; \mathcal{M}, w' \Vdash \varphi). \end{array}$$

Note the internal perspective: we evaluate formulas inside models, at particular states. Modal formulas are like little creatures that explore models by moving between related points. This is a key modal intuition, gives rise to the notion of bisimulation, and is the driving force for at least one traditional application.

Tense logic

- $\langle \mathrm{F} \rangle$ means "at some *F*uture state", and $\langle \mathrm{P} \rangle$ means "at some *P*ast state".
- $\langle \mathrm{P} \rangle$ mia-unconscious is true iff we can look back in time from the current state and see a state where Mia is unconscious. Works a bit like the sentence *Mia has been unconscious*.
- $\langle \mathrm{F} \rangle$ mia-unconscious requires us to scan the states that lie in the future looking for one where Mia is unconscious. Works a bit like the sentence *Mia will be unconscious*.

Feature logic

Consider the following Attribute Value Matrix (AVM):

$$\begin{bmatrix} \textsc{agreement} & \begin{bmatrix} \textsc{person} & \textit{1st} \\ \textsc{number} & \textit{der} \end{bmatrix} \\ \textsc{case} & -\textit{dative} \end{bmatrix}$$

Feature logic

Consider the following Attribute Value Matrix (AVM):

$$\begin{bmatrix} \textsc{agreement} & \begin{bmatrix} \textsc{person} & \textit{1st} \\ \textsc{number} & \textit{der} \end{bmatrix} \\ \textsc{case} & -\textit{dative} \end{bmatrix}$$

This is a notational variant of the following modal formula:

$$\begin{array}{ll} & \langle \textsc{agreement} \rangle (\langle \textsc{person} \rangle \text{1st} \land \langle \textsc{number} \rangle \text{singular}) \\ \land & \langle \textsc{case} \rangle \neg \text{dative} \end{array}$$

Description logic

And, moving into the heart of ordinary *extensional* logic, consider the following $\mathcal{ALC}$ term:

$$\mathsf{killer} \sqcap \exists \textsc{employer}.\mathsf{gangster}$$

Description logic

And, moving into the heart of ordinary *extensional* logic, consider the following $\mathcal{ALC}$ term:

$$\mathsf{killer} \sqcap \exists \textsc{employer}.\mathsf{gangster}$$

This means exactly the same thing as the modal formula:

$$\mathsf{killer} \land \langle \textsc{employer} \rangle \mathsf{gangster}$$

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 a Kripke model 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 it has a unary predicate symbol P. We'll use V to interpret these predicate symbols.
- For each modality $\langle \text{R} \rangle$, it has 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

$$\langle \text{F} \rangle \text{ mia} - \text{unconscious}$$

Doing it first-order style (I)

Consider the modal representation

$$\langle \text{F} \rangle \text{ mia} - \text{unconscious}$$

we could use instead the first-order representation

$$\exists t (t_o < t \land \text{MIA} - \text{UNCONSCIOUS}(t)).$$

Doing it first-order style (II)

And consider the modal representation

$$\mathsf{killer} \wedge \langle \textsc{employer} \rangle \mathsf{gangster}$$

Doing it first-order style (II)

And consider the modal representation

$$\mathsf{killer} \wedge \langle \textsc{employer} \rangle \mathsf{gangster}$$

We could use instead the first-order representation

$$\mathsf{KILLER}(x) \wedge \exists y(\mathrm{EMPLOYER}(x, y) \wedge \mathsf{GANGSTER}(y))$$

Standard Translation

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

$$\begin{array}{lcl} \textsc{st}_x(\mathrm{p}) & = & \mathrm{P}x \\ \textsc{st}_x(\neg\varphi) & = & \neg\, \textsc{st}_x(\varphi) \\ \textsc{st}_x(\varphi \wedge \psi) & = & \textsc{st}_x(\varphi) \wedge \textsc{st}_x(\psi) \\ \textsc{st}_x(\langle R \rangle \varphi) & = & \exists y(Rxy \wedge \textsc{st}_y(\varphi)) \end{array}$$

Note that $\textsc{st}_x(\varphi)$ always contains exactly one free variable (namely x).

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 $\mathcal{M} \models \textsc{st}_x(\varphi)[x \leftarrow w]$.

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.
- And as we'll later see, the reverse is not possible: first-order logic can describe models in far more detail that modal logic can. Some first-order formulas have no modal equivalent. That is, modal languages are weaker than their corresponding first-order languages.
- So why bother with modal logic?

Reasons for going modal

Reasons for going modal

- **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 a compact, easy to read, representations.

Reasons for going modal

- **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 a compact, easy to read, representations.
- **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.

Reasons for going modal

- **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 a compact, easy to read, representations.
- **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.
- **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. . .

Bisimulation (I)

Bisimulation (I)

The fundamental notion of equivalence between states for modal logic.

Bisimulations are used in other disciplines besides modal logic. Its role in all of them is to provide an appropriate notion of equivalence.

Social Network Theory Here they capture the notion of two social networks being functionally identical, even though they are not isomorphic. It's the configurations in which the agents stand in various roles that render two social networks "the same".

Theoretical Computer Science Here they embody the notion of behavioural equivalence for processes.

Non-well-founded Set Theory Here they replace the extensionality as the criterion of equality: two non-well-founded sets (graphs) are equal iff they are bisimilar.

Bisimulation (II)

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

Bisimulation (II)

Let $\mathcal{M} = (W, \mathcal{R}, V)$ and $\mathcal{M}' = (W', \mathcal{R}', V')$ be models for the same basic modal language. 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.

Bisimulation (II)

Let $\mathcal{M} = (W, \mathcal{R}, V)$ and $\mathcal{M}' = (W', \mathcal{R}', V')$ be models for the same basic modal language. 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'.

Bisimulation (II)

Let $\mathcal{M} = (W, \mathcal{R}, V)$ and $\mathcal{M}' = (W', \mathcal{R}', V')$ be models for the same basic modal language. 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'.

Modal formulas are invariant under bisimulation

Proposition: Let $\mathcal{M} = (W, \mathcal{R}, V)$ and $\mathcal{M}' = (W', \mathcal{R}', V')$ be models for the same basic modal language, 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':

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

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 (which shows that not all first-order formulas can be translated into modal formulas).
- But bisimulation invariance seems to be a natural property 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).

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, shows 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 i is marked with the information p.

and so on.

Temporal logic

- Temporal representations in Artificial Intelligence (such as Allen's system, and the situation calculus) based around temporal reference — and for good reasons.
- Worse, standard modal logics of time are completely inadequate for the temporal semantics of natural language. *Vincent accidentally squeezed the trigger* doesn't mean that at some completely unspecified past time Vincent did in fact accidentally squeeze the trigger, it means that at some *particular*, contextually determined, past time he did so. The representation, $\langle \mathrm{P} \rangle\, vincent - accidentally - squeeze - trigger$ fails to capture this.

Tense in text

Vincent woke up. Something felt very wrong. Vincent reached under his pillow for his Uzi.

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 assert the identity of the times needed for the first two sentences, nor to capture the move forward in time needed by the third.

In fact, for this reason modal languages for temporal representation have not been the tool of choice in natural language semantics for over 15 years.

Feature logic

As we've mentioned, the following Attribute Value Matrix (AVM):

$$\begin{bmatrix} \text{AGREEMENT} & \begin{bmatrix} \text{PERSON} & \textit{1st} \\ \text{NUMBER} & \textit{plural} \end{bmatrix} \\ \text{CASE} & -\textit{dative} \end{bmatrix}$$

is a notational variant of the following modal formula:

$$\begin{aligned} & \langle \text{AGREEMENT} \rangle\, (\langle \text{PERSON} \rangle\, \text{1st} \wedge \langle \text{NUMBER} \rangle\, \text{plural}) \\ \wedge \quad & \langle \text{CASE} \rangle\, \neg \text{dative} \end{aligned}$$

Feature logic

But full AVM notation is richer. It can assert re-entrancies:

$$\begin{bmatrix} \text{SUBJ} & \boxed{1} \begin{bmatrix} \text{AGR} & \textit{foo} \\ \text{PRED} & \textit{bar} \end{bmatrix} \\ \text{COMP} & [\text{SUBJ} \ \boxed{1}\,] \end{bmatrix}$$

This cannot be captured in orthodox modal logic.

Description logic I

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

$$\mathsf{killer} \sqcap \exists\text{EMPLOYER}.\mathsf{gangster}$$

$$\mathsf{killer} \wedge \langle\text{EMPLOYER}\rangle\mathsf{gangster}$$

Nonetheless, this correspondence only involves what description logicians call the TBox (Terminological Box).

Description logic II

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

$$\mathsf{mia} : \mathsf{Beautiful}$$

$$(\mathsf{jules}, \mathsf{vincent}) : \mathsf{Friends}$$

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

Ambition

- 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 will fix the obvious shortcoming — and as we shall learn, it 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). 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 called satisfaction operators.

Let's make these ideas precise ...

Syntax

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

$$\text{WFF} \quad := \quad i \mid \mathrm{p} \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi \mid \langle \mathrm{M} \rangle \varphi \mid [\mathrm{M}]\, \varphi \mid @_i\varphi$$

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∪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

$\mathcal{M}, w \Vdash a$	iff	$w \in V(a)$, where $a \in$ PROP ∪ NOM
$\mathcal{M}, w \Vdash \neg\varphi$	iff	$\mathcal{M}, w \nVdash \varphi$
$\mathcal{M}, w \Vdash \varphi \wedge \psi$	iff	$\mathcal{M}, w \Vdash \varphi$ and $\mathcal{M}, w \Vdash \psi$
$\mathcal{M}, w \Vdash \varphi \vee \psi$	iff	$\mathcal{M}, w \Vdash \varphi$ or $\mathcal{M}, w \Vdash \psi$
$\mathcal{M}, w \Vdash \varphi \rightarrow \psi$	iff	$\mathcal{M}, w \nVdash \varphi$ or $\mathcal{M}, w \Vdash \psi$
$\mathcal{M}, w \Vdash \langle \mathrm{M} \rangle \varphi$	iff	$\exists w'(wR^m w'$ & $\mathcal{M}, w' \Vdash \varphi)$
$\mathcal{M}, w \Vdash [\mathrm{M}]\, \varphi$	iff	$\forall w'(wR^m w' \;\Rightarrow\; \mathcal{M}, w' \Vdash \varphi)$.
$\mathcal{M}, w \Vdash @_i\varphi$	iff	$\mathcal{M}, \mathrm{i} \Vdash \varphi$, where i is the denotation of i under V.

Tense logic

- On the road to capturing AI temporal representation formalisms such as Allen's logic of temporal reference; @ can play the role of **Holds**.
- And we can now handle natural language examples more convincingly: $\langle \mathsf{P} \rangle (i \wedge \mathsf{Vincent\text{-}accidentally\text{-}squeeze\text{-}the\text{-}trigger})$ locates the trigger-squeezing not merely in the past, but at a specific temporal state there, namely the one named by i — capturing the meaning of *Vincent accidentally squeezed the trigger.* Let's take this a little further...

Reichenbach in hybrid logic

Structure	Name	English example	Representation
E–R–S	Pluperfect	I had seen	$\langle \mathrm{P} \rangle (i \wedge \langle \mathrm{P} \rangle \phi)$
E,R–S	Past	I saw	$\langle \mathrm{P} \rangle (i \wedge \phi)$
R–E–S	Future-in-the-past	I would see	$\langle \mathrm{P} \rangle (i \wedge \langle \mathrm{F} \rangle \phi)$
R–S,E	Future-in-the-past	I would see	$\langle \mathrm{P} \rangle (i \wedge \langle \mathrm{F} \rangle \phi)$
R–S–E	Future-in-the-past	I would see	$\langle \mathrm{P} \rangle (i \wedge \langle \mathrm{F} \rangle \phi)$
E–S,R	Perfect	I have seen	$\langle \mathrm{P} \rangle \phi$
S,R,E	Present	I see	ϕ
S,R–E	Prospective	I am going to see	$\langle \mathrm{F} \rangle \phi$
S–E–R	Future perfect	I will have seen	$\langle \mathrm{F} \rangle (i \wedge \langle \mathrm{P} \rangle \phi)$
S,E–R	Future perfect	I will have seen	$\langle \mathrm{F} \rangle (i \wedge \langle \mathrm{P} \rangle \phi)$
E–S–R	Future perfect	I will have seen	$\langle \mathrm{F} \rangle (i \wedge \langle \mathrm{P} \rangle \phi)$
S–R,E	Future	I will see	$\langle \mathrm{F} \rangle (i \wedge \phi)$
S–R–E	Future-in-the-future	(Latin: abiturus ero)	$\langle \mathrm{F} \rangle (i \wedge \langle \mathrm{F} \rangle \phi)$

Tense in text

Vincent woke up. Something felt very wrong. Vincent reached under his pillow for his Uzi.

Tense in text

Vincent woke up. Something felt very wrong. Vincent reached under his pillow for his Uzi.

$$P(i \wedge \text{vincent-wake-up})$$

Tense in text

Vincent woke up. Something felt very wrong. Vincent reached under his pillow for his Uzi.

$$P(i \wedge \text{vincent-wake-up})$$

$$\wedge \quad P(j \wedge \text{something-feel-very-wrong})$$

Tense in text

Vincent woke up. Something felt very wrong. Vincent reached under his pillow for his Uzi.

$$P(i \land \text{vincent-wake-up})$$
$$\land \quad P(j \land \text{something-feel-very-wrong}) \qquad \land \quad @_j i$$

Tense in text

Vincent woke up. Something felt very wrong. Vincent reached under his pillow for his Uzi.

$$P(i \land \text{vincent-wake-up})$$
$$\land \quad P(j \land \text{something-feel-very-wrong}) \qquad \land \quad @_j i$$
$$\land \quad P(k \land \text{vincent-reach-under-pillow-for-uzi})$$

Tense in text

Vincent woke up. Something felt very wrong. Vincent reached under his pillow for his Uzi.

$$P(i \land \text{vincent-wake-up})$$
$$\land \quad P(j \land \text{something-feel-very-wrong}) \qquad \land \quad @_j i$$
$$\land \quad P(k \land \text{vincent-reach-under-pillow-for-uzi}) \quad \land \quad @_k P i$$

Feature logic

$$\begin{bmatrix} \text{SUBJ} & \boxed{1}\begin{bmatrix} \text{AGR} & \textit{foo} \\ \text{PRED} & \textit{bar} \end{bmatrix} \\ \text{COMP} & [\text{SUBJ} \ \boxed{1}\,] \end{bmatrix}$$

This corresponds to the following hybrid wff:

$$\langle \text{SUBJ} \rangle \, (i \land \langle \text{AGR} \rangle \, \text{foo} \land \langle \text{PRED} \rangle \, \text{bar})$$
$$\land \quad \langle \text{COMP} \rangle \, \langle \text{SUBJ} \rangle \, i$$

Description logic (I)

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

$$\mathsf{mia} : \mathsf{Beautiful}$$

we can write

$$@_{\mathsf{mia}}\mathsf{Beautiful}$$

Description logic (II)

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

$$(\mathsf{jules}, \mathsf{vincent}) : \mathsf{Friends}$$

we can write

$$@_{\mathsf{jules}}\langle \mathsf{Friends} \rangle \mathsf{vincent}$$

Basic hybrid language clearly modal

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

- Nominals just atomic formulas.
- Satisfaction operators are normal modal operators. 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 effect 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 (Areces, Blackburn, and Marx).

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

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):

$$\begin{array}{lcl} \text{ST}_x(\text{p}) & = & \text{P}x \\ \text{ST}_x(i) & = & (i = x) \\ \text{ST}_x(\neg\varphi) & = & \neg\,\text{ST}_x(\varphi) \\ \text{ST}_x(\varphi \wedge \psi) & = & \text{ST}_x(\varphi) \wedge \text{ST}_x(\psi) \\ \text{ST}_x(\langle R \rangle \varphi) & = & \exists y(Rxy \wedge \text{ST}_y(\varphi)) \\ \text{ST}_x(@_i\varphi) & = & \text{ST}_i(\varphi) \end{array}$$

Note that $\text{ST}_x(\varphi)$ always contains at most 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 $\mathcal{M} \models \text{ST}_x(\varphi)[x \leftarrow w]$.

Basic hybrid logic can specify Robinson Diagrams

Basic hybrid logic can specify Robinson Diagrams

- $@_ip$ says that the states labelled i bears the information p, while $\neg @_ip$ denies this. That is, we can specify how atomic properties are distributed modally.

Basic hybrid logic can specify Robinson Diagrams

- $@_ip$ says that the states labelled i bears the information p, while $\neg @_ip$ denies this. That is, we can specify how atomic properties are distributed modally.
- $@_ij$ says that the states labelled i and j are identical, while $\neg @_ij$ says they are distinct. That is, we can specify theories of state equality modally.

Basic hybrid logic can specify Robinson Diagrams

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

Basic hybrid logic can specify Robinson Diagrams

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

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

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

Bisimulation-with-constants

Let $\mathcal{M} = (W, \mathcal{R}, V)$ and $\mathcal{M}' = (W', \mathcal{R}', V')$ be models for the same basic hybrid language. 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. All points named by nominals are related by Z.

Basic hybrid formulas are invariant under bisimulations-with-constants

Proposition: Let $\mathcal{M} = (W, \mathcal{R}, V)$ and $\mathcal{M}' = (W', \mathcal{R}', V')$ be models for the same basic hybrid language, 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':

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

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 iff (Areces, Blackburn, ten Cate, and Marx)

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.

Summing up ...

- We learned about some of the good points of orthodox modal logic, but also saw that it's 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.

Summing up ...

- We learned about some of the good points of orthodox modal logic, but also saw that it's 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 is easy to use deductively — as we shall see after a break!

Hybrid deduction

Let's continue with an example-driven introduction to hybrid deduction. We concentrate on tableau systems. We shall:

- Discuss the goals and problems of orthodox modal deduction.
- Present a hybrid tableau system for reasoning about arbitrary models.
- Show how this can be extended to hybrid tableau systems for special classes of models.
- Round off by discussing further themes in hybrid deduction, including their implementation.

Different models, different logics

Key fact about modal logic: when you work with different kinds of models (graphs) the logic typically changes. For example:

- $\Box p \wedge \Box q \rightarrow \Box(p \wedge q)$ is valid on all models: it's part of the basic, universally applicable, logic.
- But $\Diamond\Diamond p \rightarrow \Diamond p$ is only valid on transitive graphs. It's not part of the basic logic, rather it's part of the special (stronger) logic that we need to use when working with transitive models.

Modal deduction should be general

- Quite rightly, modal logicians have insisted on developing proof methods which are general — that is, which can be easily adapted to cope with the logics of many kinds of models (transitive, reflexive, symmetric, dense, and so on).
- They achieve this goal by making use of Hilbert-style systems (that is, axiomatic systems).
- There is a basic axiomatic systems (called **K**) for dealing with arbitrary models.
- To deal with special classes of models, further axioms are added to **K**. For example, adding $\Diamond\Diamond p \rightarrow \Diamond p$ as an axiom gives us the logic of transitive frames.

Generality clashes with easy of use

- Unfortunately, Hilbert systems are hard to use and completely unsuitable for computational implementation.
- For ease of use we want (say) natural deduction systems or tableau systems. For computational implementation we want (say) resolution systems or tableau systems.
- But it is hard to develop tableau, or natural deduction, or resolution in a general way in orthodox modal logic.
- Why is this?

Getting behind the diamonds

- The difficulty is extracting information from under the scope of diamonds.
- That is, given $\Diamond\varphi$, how do we lay hands on φ? And given $\neg\Box\varphi$ (that is, $\Diamond\neg\varphi$), how do we lay hands on $\neg\varphi$?
- In first order logic, the analogous problem is trivial. There is a simple rule for stripping away existential quantifiers: from $\exists x\varphi$ we conclude $\varphi[x \leftarrow a]$ for some brand new constant a (this rule is usually called Existential Elimination).
- But in orthodox modal logic there is no simple way of stripping off the diamonds.

Hybrid deduction

- Hybrid deduction is based on a simple observation: it's easy to get at the information under the scope of diamonds — for there is an natural way of stripping the diamonds away.
- We shall explore this idea in the setting of tableau — but it can (and has been) used in a variety of proof styles, including resolution and natural deduction.
- Moreover, once the tableau system for reasoning about arbitrary models has been defined, it is straightforward to extend it to to cover the logics of special classes of models. That is, hybridization enables us to achieve the traditional modal goal of generality without resorting to Hilbert-systems.

Moreover. . .

Hybrid reasoning is arguably quite natural.

In what follows we shall sometimes give an informal proof before we give the tableau proof. As we shall see, our tableau proofs mimic the informal reasoning fairly closely.

Hip and cute

Consider the following statement:

> *If everyone you hate is hip, and someone you hate is cute, then someone you hate is both hip and cute.*

We can represent it as follows:

$$[\text{HATE}]\,\text{hip} \land \langle\text{HATE}\rangle\,\text{cute} \rightarrow \langle\text{HATE}\rangle\,(\text{hip} \land \text{cute})$$

This is a valid statement, and it's validity is easy to establish informally. . .

Informal argument

Informal argument

- Suppose "If everyone you hate is hip, and someone you hate is cute, then someone you hate is both hip and cute" is not true.

Informal argument

- Suppose "If everyone you hate is hip, and someone you hate is cute, then someone you hate is both hip and cute" is not true.
- Then everyone you hate is hip, and someone you hate is cute. However no one you hate is both hip and cute.

Informal argument

- Suppose "If everyone you hate is hip, and someone you hate is cute, then someone you hate is both hip and cute" is not true.
- Then everyone you hate is hip, and someone you hate is cute. However no one you hate is both hip and cute.
- So there is someone that you hate (let's call him Jim) who is cute.

Informal argument

- Suppose "If everyone you hate is hip, and someone you hate is cute, then someone you hate is both hip and cute" is not true.
- Then everyone you hate is hip, and someone you hate is cute. However no one you hate is both hip and cute.
- So there is someone that you hate (let's call him Jim) who is cute.
- But as Jim is someone you hate, he be hip as well as cute (for everyone you hate is hip).

Informal argument

- Suppose "If everyone you hate is hip, and someone you hate is cute, then someone you hate is both hip and cute" is not true.
- Then everyone you hate is hip, and someone you hate is cute. However no one you hate is both hip and cute.
- So there is someone that you hate (let's call him Jim) who is cute.
- But as Jim is someone you hate, he be hip as well as cute (for everyone you hate is hip).
- But Jim can't be both hip and cute (for no one you hate is both hip and cute). Contradiction!. So the original statement was true after all.

$[\text{HATE}]\,\text{hip} \wedge \langle\text{HATE}\rangle\,\text{cute} \rightarrow \langle\text{HATE}\rangle\,(\text{hip} \wedge \text{cute})$

$[\text{HATE}]\,\text{hip} \wedge \langle\text{HATE}\rangle\,\text{cute} \rightarrow \langle\text{HATE}\rangle\,(\text{hip} \wedge \text{cute})$

1	$\neg @_i([\text{HATE}]\,h \wedge \langle\text{HATE}\rangle\,c \rightarrow \langle\text{HATE}\rangle\,(h \wedge c))$

$[\text{HATE}]\,\text{hip} \wedge \langle\text{HATE}\rangle\,\text{cute} \rightarrow \langle\text{HATE}\rangle\,(\text{hip} \wedge \text{cute})$

1	$\neg @_i([\text{HATE}]\,h \wedge \langle\text{HATE}\rangle\,c \rightarrow \langle\text{HATE}\rangle\,(h \wedge c))$
2	$@_i([\text{HATE}]\,h \wedge \langle\text{HATE}\rangle\,c)$
2′	$\neg @_i\langle\text{HATE}\rangle\,(h \wedge c)$

$[\text{HATE}]\,\text{hip} \wedge \langle\text{HATE}\rangle\,\text{cute} \rightarrow \langle\text{HATE}\rangle\,(\text{hip} \wedge \text{cute})$

1	$\neg @_i([\text{HATE}]\,h \wedge \langle\text{HATE}\rangle\,c \rightarrow \langle\text{HATE}\rangle\,(h \wedge c))$
2	$@_i([\text{HATE}]\,h \wedge \langle\text{HATE}\rangle\,c)$
2′	$\neg @_i\langle\text{HATE}\rangle\,(h \wedge c)$
3	$@_i[\text{HATE}]\,h$
3′	$@_i\langle\text{HATE}\rangle\,c$

$[\text{HATE}]\,\text{hip} \wedge \langle\text{HATE}\rangle\,\text{cute} \rightarrow \langle\text{HATE}\rangle\,(\text{hip} \wedge \text{cute})$

1	$\neg @_i([\text{HATE}]\,h \wedge \langle\text{HATE}\rangle\,c \rightarrow \langle\text{HATE}\rangle\,(h \wedge c))$
2	$@_i([\text{HATE}]\,h \wedge \langle\text{HATE}\rangle\,c)$
2′	$\neg @_i\langle\text{HATE}\rangle\,(h \wedge c)$
3	$@_i[\text{HATE}]\,h$
3′	$@_i\langle\text{HATE}\rangle\,c$
4	$@_i\langle\text{HATE}\rangle\,j$
4′	$@_j c$

$[\text{HATE}]\,\text{hip} \wedge \langle\text{HATE}\rangle\,\text{cute} \rightarrow \langle\text{HATE}\rangle\,(\text{hip} \wedge \text{cute})$

1 $\neg @_i([\text{HATE}]\,h \wedge \langle\text{HATE}\rangle\,c \rightarrow \langle\text{HATE}\rangle\,(h \wedge c))$
2 $@_i([\text{HATE}]\,h \wedge \langle\text{HATE}\rangle\,c)$
2′ $\neg @_i \langle\text{HATE}\rangle\,(h \wedge c)$
3 $@_i[\text{HATE}]\,h$
3′ $@_i\langle\text{HATE}\rangle\,c$
4 $@_i\langle\text{HATE}\rangle\,j$
4′ $@_j c$
5 $@_j h$

$[\text{HATE}]\,\text{hip} \wedge \langle\text{HATE}\rangle\,\text{cute} \rightarrow \langle\text{HATE}\rangle\,(\text{hip} \wedge \text{cute})$

1 $\neg @_i([\text{HATE}]\,h \wedge \langle\text{HATE}\rangle\,c \rightarrow \langle\text{HATE}\rangle\,(h \wedge c))$
2 $@_i([\text{HATE}]\,h \wedge \langle\text{HATE}\rangle\,c)$
2′ $\neg @_i \langle\text{HATE}\rangle\,(h \wedge c)$
3 $@_i[\text{HATE}]\,h$
3′ $@_i\langle\text{HATE}\rangle\,c$
4 $@_i\langle\text{HATE}\rangle\,j$
4′ $@_j c$
5 $@_j h$
6 $\neg @_j (h \wedge c)$

$[\text{HATE}]\,\text{hip} \wedge \langle\text{HATE}\rangle\,\text{cute} \rightarrow \langle\text{HATE}\rangle\,(\text{hip} \wedge \text{cute})$

1 $\neg @_i([\text{HATE}]\,h \wedge \langle\text{HATE}\rangle\,c \rightarrow \langle\text{HATE}\rangle\,(h \wedge c))$
2 $@_i([\text{HATE}]\,h \wedge \langle\text{HATE}\rangle\,c)$
2′ $\neg @_i \langle\text{HATE}\rangle\,(h \wedge c)$
3 $@_i[\text{HATE}]\,h$
3′ $@_i\langle\text{HATE}\rangle\,c$
4 $@_i\langle\text{HATE}\rangle\,j$
4′ $@_j c$
5 $@_j h$
6 $\neg @_j (h \wedge c)$
7 $\neg @_j h$ | $\neg @_j c$

$[\text{HATE}]\,\text{hip} \wedge \langle\text{HATE}\rangle\,\text{cute} \rightarrow \langle\text{HATE}\rangle\,(\text{hip} \wedge \text{cute})$

1 $\neg @_i([\text{HATE}]\,h \wedge \langle\text{HATE}\rangle\,c \rightarrow \langle\text{HATE}\rangle\,(h \wedge c))$
2 $@_i([\text{HATE}]\,h \wedge \langle\text{HATE}\rangle\,c)$
2′ $\neg @_i \langle\text{HATE}\rangle\,(h \wedge c)$
3 $@_i[\text{HATE}]\,h$
3′ $@_i\langle\text{HATE}\rangle\,c$
4 $@_i\langle\text{HATE}\rangle\,j$
4′ $@_j c$
5 $@_j h$
6 $\neg @_j (h \wedge c)$
7 $\neg @_j h$ | $\neg @_j c$
$\bot_{5,7}$

$[\text{HATE}]\,\text{hip} \wedge \langle\text{HATE}\rangle\,\text{cute} \rightarrow \langle\text{HATE}\rangle\,(\text{hip} \wedge \text{cute})$

1 $\neg @_i([\text{HATE}]\,h \wedge \langle\text{HATE}\rangle\,c \rightarrow \langle\text{HATE}\rangle\,(h \wedge c))$
2 $@_i([\text{HATE}]\,h \wedge \langle\text{HATE}\rangle\,c)$
2′ $\neg @_i \langle\text{HATE}\rangle\,(h \wedge c)$
3 $@_i[\text{HATE}]\,h$
3′ $@_i\langle\text{HATE}\rangle\,c$
4 $@_i\langle\text{HATE}\rangle\,j$
4′ $@_j c$
5 $@_j h$
6 $\neg @_j (h \wedge c)$
7 $\neg @_j h$ | $\neg @_j c$
$\bot_{5,7}$ | $\bot_{4',7}$

Internalizing Labelled Deduction

$\neg$ rules $\quad \dfrac{@_i\neg\varphi}{\neg @_i\varphi} \qquad \dfrac{\neg @_i\neg\varphi}{@_i\varphi}$

$\wedge$ rules $\quad \dfrac{@_i(\varphi \wedge \psi)}{@_i\varphi \quad @_i\psi} \qquad \dfrac{\neg @_i(\varphi \wedge \psi)}{\neg @_i\varphi \mid \neg @_i\psi}$

$@$ rules $\quad \dfrac{@_i @_j\varphi}{@_j\varphi} \qquad \dfrac{\neg @_i @_j\varphi}{\neg @_j\varphi}$

Extracting information from modal contexts

In the statement of these rules we write j to indicate a nominal new to the branch where the rule is being applied.

Extracting information from modal contexts

In the statement of these rules we write j to indicate a nominal new to the branch where the rule is being applied.

$\Diamond$ rules $\quad \dfrac{@_i\langle R\rangle\,\varphi}{@_i\langle R\rangle\,j \quad @_j\varphi} \qquad \dfrac{\neg @_i\langle R\rangle\,\varphi \quad @_i\langle R\rangle\,k}{\neg @_k\varphi}$

Extracting information from modal contexts

In the statement of these rules we write j to indicate a nominal new to the branch where the rule is being applied.

$\Diamond$ rules

$$\frac{@_i\langle R\rangle\varphi}{@_i\langle R\rangle j \quad @_j\varphi} \qquad \frac{\neg @_i\langle R\rangle\varphi \quad @_i\langle R\rangle k}{\neg @_k\varphi}$$

$\Box$ rules

$$\frac{@_i[R]\varphi \quad @_i\langle R\rangle k}{@_k\varphi} \qquad \frac{\neg @_i[R]\varphi}{@_i\langle R\rangle j \quad \neg @_j\varphi}$$

Link with first-order deduction (Studio Version)

Link with first-order deduction (Studio Version)

- The hybrid rule from $@_i\Diamond\varphi$ conclude $@_i\Diamond j$ and $@_j\varphi$ is essentially the first-order rule of Existential Elimination (from $\exists x\varphi$ conclude $\varphi[x \leftarrow j]$).

Link with first-order deduction (Studio Version)

- The hybrid rule from $@_i\Diamond\varphi$ conclude $@_i\Diamond j$ and $@_j\varphi$ is essentially the first-order rule of Existential Elimination (from $\exists x\varphi$ conclude $\varphi[x \leftarrow j]$).
- Recall that (via the Standard Translation) we know that $\Diamond\varphi$ is shorthand for $\exists y(Riy \wedge \text{ST}_y(\varphi))$.

Link with first-order deduction (Studio Version)

- The hybrid rule from $@_i\Diamond\varphi$ conclude $@_i\Diamond j$ and $@_j\varphi$ is essentially the first-order rule of Existential Elimination (from $\exists x\varphi$ conclude $\varphi[x \leftarrow j]$).
- Recall that (via the Standard Translation) we know that $\Diamond\varphi$ is shorthand for $\exists y(Riy \wedge \text{ST}_y(\varphi))$.
- Applying Existential Elimination to this yields $Rij \wedge \text{ST}_j(\varphi)$. But this is just $@_i\Diamond j \wedge @_j\varphi$, the output of the tableau rule.

Link with first-order deduction (Studio Version)

- The hybrid rule from $@_i\Diamond\varphi$ conclude $@_i\Diamond j$ and $@_j\varphi$ is essentially the first-order rule of Existential Elimination (from $\exists x\varphi$ conclude $\varphi[x \leftarrow j]$).
- Recall that (via the Standard Translation) we know that $\Diamond\varphi$ is shorthand for $\exists y(Riy \wedge \text{ST}_y(\varphi))$.
- Applying Existential Elimination to this yields $Rij \wedge \text{ST}_j(\varphi)$. But this is just $@_i\Diamond j \wedge @_j\varphi$, the output of the tableau rule.
- In short, nominals give us exactly the grip we need on the bound variables hidden by modal notation. They give us the benefits of first-order techniques in a decidable logic.

Link with first-order deduction (Live Version)

Hybrid Logic	First Order Logic
$@_i\Diamond\phi$	

Link with first-order deduction (Live Version)

Hybrid Logic	First Order Logic
$@_i\Diamond\phi$	$\exists y(Riy \wedge ST_y(\phi))$

Link with first-order deduction (Live Version)

Hybrid Logic	First Order Logic
$@_i\Diamond\phi$	$\exists y(Riy \wedge ST_y(\phi))$
$@_i\Diamond j$	
$@_j\phi$	

Link with first-order deduction (Live Version)

Hybrid Logic	First Order Logic
$@_i\Diamond\phi$	$\exists y(Riy \wedge ST_y(\phi))$
	$Rij \wedge ST_j(\phi)$
$@_i\Diamond j$	
$@_j\phi$	

Link with first-order deduction (Live Version)

Hybrid Logic	First Order Logic
$@_i\Diamond\phi$	$\exists y(Riy \wedge ST_y(\phi))$
	$Rij \wedge ST_j(\phi)$
$@_i\Diamond j$	Rij
$@_j\phi$	$ST_j(\phi)$

Equality rules

But more rules are needed. Why? Nothing we have said so far gets to grips with fact that nominals have an intrinsic logic. Nominals give us a modal theory of equality, and we need to get to deal with this. Here's one way of doing this:

$$\frac{(i \text{ occurs on branch})}{@_i i} \qquad \frac{@_i j \quad @_i\varphi}{@_j\varphi} \qquad \frac{@_i\Diamond j \quad @_j k}{@_i\Diamond k}$$

$(\Diamond p \wedge \Diamond\neg p) \rightarrow (\Box(q \rightarrow i) \rightarrow \Diamond\neg q)$

$(\Diamond p \wedge \Diamond\neg p) \rightarrow (\Box(q \rightarrow i) \rightarrow \Diamond\neg q)$

1	$\neg @_i((\Diamond p \wedge \Diamond\neg p) \rightarrow (\Box(q \rightarrow i) \rightarrow \Diamond\neg q))$	

$(\Diamond p \wedge \Diamond\neg p) \rightarrow (\Box(q \rightarrow i) \rightarrow \Diamond\neg q)$

1	$\neg @_i((\Diamond p \wedge \Diamond\neg p) \rightarrow (\Box(q \rightarrow i) \rightarrow \Diamond\neg q))$	
2	$@_i(\Diamond p \wedge \Diamond\neg p)$	
2′	$\neg @_i(\Box(q \rightarrow i) \rightarrow \Diamond\neg q)$	Propositional rule on 1

$(\Diamond p \wedge \Diamond\neg p) \rightarrow (\Box(q \rightarrow i) \rightarrow \Diamond\neg q)$

1	$\neg @_i((\Diamond p \wedge \Diamond\neg p) \rightarrow (\Box(q \rightarrow i) \rightarrow \Diamond\neg q))$	
2	$@_i(\Diamond p \wedge \Diamond\neg p)$	
2′	$\neg @_i(\Box(q \rightarrow i) \rightarrow \Diamond\neg q)$	Propositional rule on 1
3	$@_i\Diamond p$	
3′	$@_i\Diamond\neg p$	Propositional rule on 2

$(\Diamond p \wedge \Diamond \neg p) \to (\Box(q \to i) \to \Diamond \neg q)$

1	$\neg @_i((\Diamond p \wedge \Diamond \neg p) \to (\Box(q \to i) \to \Diamond \neg q))$	
2	$@_i(\Diamond p \wedge \Diamond \neg p)$	
2′	$\neg @_i(\Box(q \to i) \to \Diamond \neg q)$	Propositional rule on 1
3	$@_i \Diamond p$	
3′	$@_i \Diamond \neg p$	Propositional rule on 2
4	$@_i \Diamond j$	
4′	$@_j p$	$\Diamond$ rule on 3

$(\Diamond p \wedge \Diamond \neg p) \to (\Box(q \to i) \to \Diamond \neg q)$

1	$\neg @_i((\Diamond p \wedge \Diamond \neg p) \to (\Box(q \to i) \to \Diamond \neg q))$	
2	$@_i(\Diamond p \wedge \Diamond \neg p)$	
2′	$\neg @_i(\Box(q \to i) \to \Diamond \neg q)$	Propositional rule on 1
3	$@_i \Diamond p$	
3′	$@_i \Diamond \neg p$	Propositional rule on 2
4	$@_i \Diamond j$	
4′	$@_j p$	$\Diamond$ rule on 3
5	$@_i \Diamond k$	
5′	$@_k \neg p$	$\Diamond$ rule on 3’

$(\Diamond p \wedge \Diamond \neg p) \to (\Box(q \to i) \to \Diamond \neg q)$

1	$\neg @_i((\Diamond p \wedge \Diamond \neg p) \to (\Box(q \to i) \to \Diamond \neg q))$	
2	$@_i(\Diamond p \wedge \Diamond \neg p)$	
2′	$\neg @_i(\Box(q \to i) \to \Diamond \neg q)$	Propositional rule on 1
3	$@_i \Diamond p$	
3′	$@_i \Diamond \neg p$	Propositional rule on 2
4	$@_i \Diamond j$	
4′	$@_j p$	$\Diamond$ rule on 3
5	$@_i \Diamond k$	
5′	$@_k \neg p$	$\Diamond$ rule on 3’
6	$@_i \Box(q \to i)$	
6′	$\neg @_i \Diamond \neg q$	Propositional rule on 2’

The proof continued…

4	$@_i \Diamond j$	
4′	$@_j p$	
5	$@_i \Diamond k$	
5′	$@_k \neg p$	
6	$@_i \Box(q \to i)$	
6′	$\neg @_i \Diamond \neg q$	

The proof continued…

4	$@_i \Diamond j$	
4′	$@_j p$	
5	$@_i \Diamond k$	
5′	$@_k \neg p$	
6	$@_i \Box(q \to i)$	
6′	$\neg @_i \Diamond \neg q$	
7	$@_j q$	$\neg\Diamond$ rule on 4 and 6’, then $\neg @$ rule

The proof continued…

4	$@_i \Diamond j$	
4′	$@_j p$	
5	$@_i \Diamond k$	
5′	$@_k \neg p$	
6	$@_i \Box(q \to i)$	
6′	$\neg @_i \Diamond \neg q$	
7	$@_j q$	$\neg\Diamond$ rule on 4 and 6’, then $\neg @$ rule
8	$@_j(q \to i)$	$\Box$ rule on 4 and 6

The proof continued…

4	$@_i \Diamond j$		
4′	$@_j p$		
5	$@_i \Diamond k$		
5′	$@_k \neg p$		
6	$@_i \Box(q \to i)$		
6′	$\neg @_i \Diamond \neg q$		
7	$@_j q$		$\neg\Diamond$ rule on 4 and 6’, then $\neg @$ rule
8	$@_j(q \to i)$		$\Box$ rule on 4 and 6
9	$\neg @_j q$	$@_j i$	Propositional rule on 7 and 8
	$\bot_{7,9}$		

The proof continued…

4	$@_i \Diamond j$		
4′	$@_j p$		
5	$@_i \Diamond k$		
5′	$@_k \neg p$		
6	$@_i \Box(q \to i)$		
6′	$\neg @_i \Diamond \neg q$		
7	$@_j q$		$\neg\Diamond$ rule on 4 and 6’, then $\neg @$ rule
8	$@_j(q \to i)$		$\Box$ rule on 4 and 6
9	$\neg @_j q$	$@_j i$	Propositional rule on 7 and 8
	$\bot_{7,9}$		

The proof continued...

4	$@_i\Diamond j$	
4′	$@_j p$	
5	$@_i\Diamond k$	
5′	$@_k\neg p$	
6	$@_i\Box(q\to i)$	
6′	$\neg @_i\Diamond\neg q$	
9	$@_j i$	

The proof continued...

4	$@_i\Diamond j$	
4′	$@_j p$	
5	$@_i\Diamond k$	
5′	$@_k\neg p$	
6	$@_i\Box(q\to i)$	
6′	$\neg @_i\Diamond\neg q$	
9	$@_j i$	
10	$@_k q$	$\neg\Diamond$ rule on 5 and 6', then $\neg @$ rule

The proof continued...

4	$@_i\Diamond j$	
4′	$@_j p$	
5	$@_i\Diamond k$	
5′	$@_k\neg p$	
6	$@_i\Box(q\to i)$	
6′	$\neg @_i\Diamond\neg q$	
9	$@_j i$	
10	$@_k q$	$\neg\Diamond$ rule on 5 and 6', then $\neg @$ rule
11	$@_k(q\to i)$	$\Box$ rule on 5 and 6

The proof continued...

4	$@_i\Diamond j$	
4′	$@_j p$	
5	$@_i\Diamond k$	
5′	$@_k\neg p$	
6	$@_i\Box(q\to i)$	
6′	$\neg @_i\Diamond\neg q$	
9	$@_j i$	
10	$@_k q$	$\neg\Diamond$ rule on 5 and 6', then $\neg @$ rule
11	$@_k(q\to i)$	$\Box$ rule on 5 and 6
12	$@_k i$	Modus Ponens on 10 and 11

The proof continued...

4	$@_i\Diamond j$	
4′	$@_j p$	
5	$@_i\Diamond k$	
5′	$@_k\neg p$	
6	$@_i\Box(q\to i)$	
6′	$\neg @_i\Diamond\neg q$	
9	$@_j i$	
10	$@_k q$	$\neg\Diamond$ rule on 5 and 6', then $\neg @$ rule
11	$@_k(q\to i)$	$\Box$ rule on 5 and 6
12	$@_k i$	Modus Ponens on 10 and 11
13	$@_i p$	Nom on 4' and 9

The proof continued...

4	$@_i\Diamond j$	
4′	$@_j p$	
5	$@_i\Diamond k$	
5′	$@_k\neg p$	
6	$@_i\Box(q\to i)$	
6′	$\neg @_i\Diamond\neg q$	
9	$@_j i$	
10	$@_k q$	$\neg\Diamond$ rule on 5 and 6', then $\neg @$ rule
11	$@_k(q\to i)$	$\Box$ rule on 5 and 6
12	$@_k i$	Modus Ponens on 10 and 11
13	$@_i p$	Nom on 4' and 9
14	$@_i\neg p$	Nom on 5' and 12

The proof continued...

4	$@_i\Diamond j$	
4′	$@_j p$	
5	$@_i\Diamond k$	
5′	$@_k\neg p$	
6	$@_i\Box(q\to i)$	
6′	$\neg @_i\Diamond\neg q$	
9	$@_j i$	
10	$@_k q$	$\neg\Diamond$ rule on 5 and 6', then $\neg @$ rule
11	$@_k(q\to i)$	$\Box$ rule on 5 and 6
12	$@_k i$	Modus Ponens on 10 and 11
13	$@_i p$	Nom on 4' and 9
14	$@_i\neg p$	Nom on 5' and 12
15	$\neg @_i p$	$\neg$ rule on 14 — contradiction!

Reasoning over other classes of models

- Our tableau system deals (correctly and completely) with reasoning over arbitrary models, that is, models where we have made no special assumptions about the underlying relations. For some applications this is sufficient.
- But (as we said at the start of the lecture) in many applications we are interested in models where the relations interpreting the modalities have special properties, such as symmetry, transitivity, irreflexivity, density, discreteness, antisymmetry, determinism, and so on. We need to find a way of coping with such frame conditions in hybrid logic.
- Our basic tableau system cannot handle such requirements — but it can be easily extended to cope with them, thus meeting the traditional modal goal of generality. We'll look at two examples.

Nice neighbours

Consider the following statement:

If you have a neighbour who only has nice neighbours, then you are nice.

We can represent it as follows:

$$\langle \textsc{neighbour} \rangle\, [\textsc{neighbour}]\, \text{nice} \rightarrow \text{nice}$$

This is true no matter how the adjective "nice" is interpreted. Its truth hinges on the fact that neighbourhood is a symmetric relation.

Informal Argument

Informal Argument

- Suppose ⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice is false of you.

Informal Argument

- Suppose ⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice is false of you.
- Then ⟨NEIGHBOUR⟩ [NEIGHBOUR] nice is true of you, but nice is false of you (that is, you are not nice).

Informal Argument

- Suppose ⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice is false of you.
- Then ⟨NEIGHBOUR⟩ [NEIGHBOUR] nice is true of you, but nice is false of you (that is, you are not nice).

Informal Argument

- Suppose ⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice is false of you.
- Then ⟨NEIGHBOUR⟩ [NEIGHBOUR] nice is true of you, but nice is false of you (that is, you are not nice).
- Then you have a neighbour (let's call him Joe) who only has nice neighbours (that is, [NEIGHBOUR] nice is true of Joe).

Informal Argument

- Suppose ⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice is false of you.
- Then ⟨NEIGHBOUR⟩ [NEIGHBOUR] nice is true of you, but nice is false of you (that is, you are not nice).
- Then you have a neighbour (let's call him Joe) who only has nice neighbours (that is, [NEIGHBOUR] nice is true of Joe).
- But neighbourhood is a symmetric relation, hence you are one of Joe's neighbours.

Informal Argument

- Suppose ⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice is false of you.
- Then ⟨NEIGHBOUR⟩ [NEIGHBOUR] nice is true of you, but nice is false of you (that is, you are not nice).
- Then you have a neighbour (let's call him Joe) who only has nice neighbours (that is, [NEIGHBOUR] nice is true of Joe).
- But neighbourhood is a symmetric relation, hence you are one of Joe's neighbours.
- But all Joe's neighbours are nice — so you must be nice too. Contradiction!

Informal Argument

- Suppose ⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice is false of you.
- Then ⟨NEIGHBOUR⟩ [NEIGHBOUR] nice is true of you, but nice is false of you (that is, you are not nice).
- Then you have a neighbour (let's call him Joe) who only has nice neighbours (that is, [NEIGHBOUR] nice is true of Joe).
- But neighbourhood is a symmetric relation, hence you are one of Joe's neighbours.
- But all Joe's neighbours are nice — so you must be nice too. Contradiction!
- So ⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice must true of you after all.

Informal Argument

- Suppose ⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice is false of you.
- Then ⟨NEIGHBOUR⟩ [NEIGHBOUR] nice is true of you, but nice is false of you (that is, you are not nice).
- Then you have a neighbour (let's call him Joe) who only has nice neighbours (that is, [NEIGHBOUR] nice is true of Joe).
- But neighbourhood is a symmetric relation, hence you are one of Joe's neighbours.
- But all Joe's neighbours are nice — so you must be nice too. Contradiction!
- So ⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice must true of you after all.

But can we mimic this argument using our existing tableau system? Let's try...

⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice

⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice

1	$@_i$(⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice)	

⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice

1	$@_i$(⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice)	
2	$@_i$⟨NEIGHBOUR⟩ [NEIGHBOUR] nice	
2′	$\neg @_i$nice	Propositional rule on 1

⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice

1	$@_i$(⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice)	
2	$@_i$⟨NEIGHBOUR⟩ [NEIGHBOUR] nice	
2′	$\neg @_i$nice	Propositional rule on 1
3	$@_i$⟨NEIGHBOUR⟩ j	
3′	$@_j$[NEIGHBOUR] nice	$\Diamond$ rule on 2

⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice

1	$@_i$(⟨NEIGHBOUR⟩ [NEIGHBOUR] nice → nice)	
2	$@_i$⟨NEIGHBOUR⟩ [NEIGHBOUR] nice	
2′	$\neg @_i$nice	Propositional rule on 1
3	$@_i$⟨NEIGHBOUR⟩ j	
3′	$@_j$[NEIGHBOUR] nice	$\Diamond$ rule on 2

Now we are blocked. There is no way to close this branch.

But there is an easy solution

Add the following rule when working with symmetric relations:

$$\frac{@_i\langle \text{NEIGHBOUR}\rangle\, j}{@_j\langle \text{NEIGHBOUR}\rangle\, i}$$

(Here i and j can be any nominals on the branch we are working on).
This rule is a direct expression of symmetry, and with its help we can finish off our proof.

$\langle\text{NEIGHBOUR}\rangle\, [\text{NEIGHBOUR}]\,\text{nice} \rightarrow \text{nice}$

1	$@_i(\langle\text{NEIGHBOUR}\rangle\, [\text{NEIGHBOUR}]\,\text{nice} \rightarrow \text{nice})$	
2	$@_i\langle\text{NEIGHBOUR}\rangle\, [\text{NEIGHBOUR}]\,\text{nice}$	
2′	$\neg @_i\text{nice}$	Propositional rule on 1
3	$@_i\langle\text{NEIGHBOUR}\rangle\, j$	
3′	$@_j[\text{NEIGHBOUR}]\,\text{nice}$	◊ rule on 2

$\langle\text{NEIGHBOUR}\rangle\, [\text{NEIGHBOUR}]\,\text{nice} \rightarrow \text{nice}$

1	$@_i(\langle\text{NEIGHBOUR}\rangle\, [\text{NEIGHBOUR}]\,\text{nice} \rightarrow \text{nice})$	
2	$@_i\langle\text{NEIGHBOUR}\rangle\, [\text{NEIGHBOUR}]\,\text{nice}$	
2′	$\neg @_i\text{nice}$	Propositional rule on 1
3	$@_i\langle\text{NEIGHBOUR}\rangle\, j$	
3′	$@_j[\text{NEIGHBOUR}]\,\text{nice}$	◊ rule on 2
4	$@_j\langle\text{NEIGHBOUR}\rangle\, i$	Symmetry rule on 3

$\langle\text{NEIGHBOUR}\rangle\, [\text{NEIGHBOUR}]\,\text{nice} \rightarrow \text{nice}$

1	$@_i(\langle\text{NEIGHBOUR}\rangle\, [\text{NEIGHBOUR}]\,\text{nice} \rightarrow \text{nice})$	
2	$@_i\langle\text{NEIGHBOUR}\rangle\, [\text{NEIGHBOUR}]\,\text{nice}$	
2′	$\neg @_i\text{nice}$	Propositional rule on 1
3	$@_i\langle\text{NEIGHBOUR}\rangle\, j$	
3′	$@_j[\text{NEIGHBOUR}]\,\text{nice}$	◊ rule on 2
4	$@_j\langle\text{NEIGHBOUR}\rangle\, i$	Symmetry rule on 3
5	$@_i\text{nice}$	□ rule on 3’ and 4

$\langle\text{NEIGHBOUR}\rangle\, [\text{NEIGHBOUR}]\,\text{nice} \rightarrow \text{nice}$

1	$@_i(\langle\text{NEIGHBOUR}\rangle\, [\text{NEIGHBOUR}]\,\text{nice} \rightarrow \text{nice})$	
2	$@_i\langle\text{NEIGHBOUR}\rangle\, [\text{NEIGHBOUR}]\,\text{nice}$	
2′	$\neg @_i\text{nice}$	Propositional rule on 1
3	$@_i\langle\text{NEIGHBOUR}\rangle\, j$	
3′	$@_j[\text{NEIGHBOUR}]\,\text{nice}$	◊ rule on 2
4	$@_j\langle\text{NEIGHBOUR}\rangle\, i$	Symmetry rule on 3
5	$@_i\text{nice}$	□ rule on 3’ and 4
	$\bot_{2',5}$	

Loop-free time

Consider the following statement:

> *If time i precedes time j, then time j does not precede time i.*

We can represent the statement as follows (where $\langle \text{F} \rangle$ is a diamond meaning “sometime-in-the-future”):

$$@_i\langle \text{F}\rangle\, j \rightarrow \neg @_j\langle \text{F}\rangle\, i$$

If you accept that temporal precedence is both transitive and irreflexive (the usual assumption) then this is a valid statement.

Informal Argument

Informal Argument

- Suppose that “if i precedes time j, then time j does not precede time i” is false.

Informal Argument

- Suppose that “if i precedes time j, then time j does not precede time i” is false.
- Then time i precedes time j, but time j precedes time i too.

Informal Argument

- Suppose that "if i precedes time j, then time j does not precede time i" is false.
- Then time i precedes time j, but time j precedes time i too.
- But temporal precedence is transitive, so time i precedes time i.

Informal Argument

- Suppose that "if i precedes time j, then time j does not precede time i" is false.
- Then time i precedes time j, but time j precedes time i too.
- But temporal precedence is transitive, so time i precedes time i.
- But temporal precedence is irreflexive, so time i cannot precede time i.

Informal Argument

- Suppose that "if i precedes time j, then time j does not precede time i" is false.
- Then time i precedes time j, but time j precedes time i too.
- But temporal precedence is transitive, so time i precedes time i.
- But temporal precedence is irreflexive, so time i cannot precede time i.
- From this contradiction we conclude that our original statement was true after all.

But can we prove $@_i\langle F\rangle j \rightarrow \neg @_j\langle F\rangle i$ using our existing tableau system? Let's try...

But can we prove $@_i\langle F\rangle j \rightarrow \neg @_j\langle F\rangle i$ using our existing tableau system? Let's try...

1	$\neg @_k(@_i\langle F\rangle j \rightarrow \neg @_j\langle F\rangle i)$	

But can we prove $@_i\langle F\rangle j \rightarrow \neg @_j\langle F\rangle i$ using our existing tableau system? Let's try...

1	$\neg @_k(@_i\langle F\rangle j \rightarrow \neg @_j\langle F\rangle i)$	
2	$@_k @_i\langle F\rangle j$	
2′	$\neg @_k \neg @_j\langle F\rangle i)$	Propositional rule on 1

But can we prove $@_i\langle F\rangle j \rightarrow \neg @_j\langle F\rangle i$ using our existing tableau system? Let's try...

1	$\neg @_k(@_i\langle F\rangle j \rightarrow \neg @_j\langle F\rangle i)$	
2	$@_k @_i\langle F\rangle j$	
2′	$\neg @_k \neg @_j\langle F\rangle i)$	Propositional rule on 1
3	$@_i\langle F\rangle j$	@ rule on 2

But can we prove $@_i\langle F\rangle j \rightarrow \neg @_j\langle F\rangle i$ using our existing tableau system? Let's try...

1	$\neg @_k(@_i\langle F\rangle j \rightarrow \neg @_j\langle F\rangle i)$	
2	$@_k @_i\langle F\rangle j$	
2′	$\neg @_k \neg @_j\langle F\rangle i)$	Propositional rule on 1
3	$@_i\langle F\rangle j$	@ rule on 2
4	$@_j\langle F\rangle i$	¬@¬ rule on 2’

But can we prove $@_i\langle F\rangle j \to \neg @_j\langle F\rangle i$ using our existing tableau system? Let's try…

1	$\neg @_k(@_i\langle F\rangle j \to \neg @_j\langle F\rangle i)$	
2	$@_k @_i\langle F\rangle j$	
2′	$\neg @_k \neg @_j\langle F\rangle i)$	Propositional rule on 1
3	$@_i\langle F\rangle j$	@ rule on 2
4	$@_j\langle F\rangle i$	¬@¬ rule on 2'

Now we are blocked. There is no way to close this branch.

But there is an easy solution

Add the following rules when working with irreflexive and transitive relations:

$$\frac{}{@_i \neg\langle F\rangle i} \qquad \frac{@_i\langle F\rangle j \quad @_j\langle F\rangle k}{@_i\langle F\rangle k}$$

(Here i, j and k can be any nominals on the branch we are working on).
These rules are a direct expression of irreflexivity and transitivity, and with their help we can finish off our proof.

$@_i\langle F\rangle j \to \neg @_j\langle F\rangle i$

1	$\neg @_k(@_i\langle F\rangle j \to \neg @_j\langle F\rangle i)$	
2	$@_k @_i\langle F\rangle j$	
2′	$\neg @_k \neg @_j\langle F\rangle i)$	Propositional rule on 1
3	$@_i\langle F\rangle j$	@ rule on 2
4	$@_j\langle F\rangle i$	¬@¬ rule on 2'

$@_i\langle F\rangle j \to \neg @_j\langle F\rangle i$

1	$\neg @_k(@_i\langle F\rangle j \to \neg @_j\langle F\rangle i)$	
2	$@_k @_i\langle F\rangle j$	
2′	$\neg @_k \neg @_j\langle F\rangle i)$	Propositional rule on 1
3	$@_i\langle F\rangle j$	@ rule on 2
4	$@_j\langle F\rangle i$	¬@¬ rule on 2'
5	$@_i\langle F\rangle i$	Transitivity rule on 3 and 4

$@_i\langle F\rangle j \to \neg @_j\langle F\rangle i$

1	$\neg @_k(@_i\langle F\rangle j \to \neg @_j\langle F\rangle i)$	
2	$@_k @_i\langle F\rangle j$	
2′	$\neg @_k \neg @_j\langle F\rangle i)$	Propositional rule on 1
3	$@_i\langle F\rangle j$	@ rule on 2
4	$@_j\langle F\rangle i$	¬@¬ rule on 2'
5	$@_i\langle F\rangle i$	Transitivity rule on 3 and 4
6	$\neg @_i\langle F\rangle i$	Irreflexivity rule

$@_i\langle F\rangle j \to \neg @_j\langle F\rangle i$

1	$\neg @_k(@_i\langle F\rangle j \to \neg @_j\langle F\rangle i)$	
2	$@_k @_i\langle F\rangle j$	
2′	$\neg @_k \neg @_j\langle F\rangle i)$	Propositional rule on 1
3	$@_i\langle F\rangle j$	@ rule on 2
4	$@_j\langle F\rangle i$	¬@¬ rule on 2'
5	$@_i\langle F\rangle i$	Transitivity rule on 3 and 4
6	$\neg @_i\langle F\rangle i$	Irreflexivity rule
	$\bot_{5,6}$	

Pure formulas

- It's time to be more precise about what completeness results are possible here.
- To do this we need to think about 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).
- We'll first discuss what we can say about frames using pure formulas, and then we'll state a general result about how they can help us in hybrid deduction.

Frame definability (I)

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

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

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*

From formulas to tableau rules

Let $@_i\varphi$ be a pure formula, built out of nominals $i, i_1, \ldots, i_n$. Then the simplest (though not always the smartest!) way of turning this formula into a tableau rule is as follows:

$$\frac{(j, j_1, \ldots, j_n \text{ on branch})}{@_i\varphi[i \leftarrow j, i_1 \leftarrow j_1, \ldots, i_n \leftarrow j_n]}$$

This rule simply says: for any branch B of the tableau you are building, you are free to instantiate $@_i\varphi$ with nominals occurring on B and add the resulting formula to the end of B.

Frame definability and deduction match for pure formulas

Completeness Theorem Suppose you extend the basic tableau system with the tableau rules for the pure formulas $@_j\varphi$, ..., $@_k\psi$ (that is, the rules of the form just described). Then the resulting system is (sound and) complete with respect to the class of frames defined by these formulas.

That is, the frame-defining and deductive powers of pure formulas match perfectly for pure formulas.

Two comments should be made about this result...

We can use any pure formula

At first glance, it seems that this completeness result only covers pure formulas of the form $@_i\varphi$. But many interesting pure formulas are not of this form. For example symmetry: $@_i\Diamond j \rightarrow @_j\Diamond i$.

Note, however, that for any pure formula φ, and any nominals i, φ and $@_i\varphi$ define exactly the same class of frames.

For example symmetry can be defined by $@_k(@_i\Diamond j \rightarrow @_j\Diamond i)$.

So our completeness theorem is fully general: it covers all classes of frames definable by a pure formulas.

But we can often be smarter

Suppose we want a complete system for symmetry. We could do this by adding the rule suggested by the previous system:

$$\frac{}{@_k(@_i\Diamond j \rightarrow @_j\Diamond i)}.$$

But in the nice neighbours example we used the following rule instead:

$$\frac{@_i\Diamond j}{@_j\Diamond i}$$

This rule is smarter: it saves us having to use tableau rules to get rid of the outermost $@_k$, and then break down the implication.

Slightly more generally

Given a pure formula of the form

$$(@_i\varphi_1 \wedge \cdots \wedge @_j\varphi_n) \rightarrow (@_k\varphi_{n+1} \vee \cdots \vee @_l\varphi_{n+m})$$

we can turn it into the tableau rule

$$\frac{@_i\varphi_1, \ldots, @_j\varphi_n}{@_k\varphi_{n+1} \mid \cdots \mid @_l\varphi_{n+m}}$$

without losing completeness.

Further themes in hybrid deduction

To conclude, let's briefly address the following questions:

- Why are general completeness proof easy to come by in hybrid logic?
- Can we really adapt these ideas to other proof styles?
- Is any of this stuff implementable?

Why are general completeness proofs so easy to come by in hybrid logic?

- Essentially because the basic hybrid logic enables us to use first-order techniques to build models.
- For example, when proving completeness for hybrid Hilbert systems, it's not necessary to use modal-style canonical models — you can build what are basically first-order Henkin models.
- And for tableau completeness proofs, observe that the tableau rules crunch formulas down into expressions of the form $(\neg)@_i\text{p}$, $(\neg)@_i j$ and $(\neg)@_i\Diamond j$. Open branches are thus Robinson diagrams of satisfying models.

Named models are important

- Moreover, the models we build in this way are named. (A named model is a model in which every point is named by some nominal.)
- A simple model theoretic argument shows that if all instances of a pure formula φ are true at all states in a named model, then the underlying frame validates ϕ. This gives us completeness for any extension by pure axioms.

Can we really adapt these ideas to other proof styles?

Yes. The key insight is that the combination of nominals and @ allow us to extract information from behind the scope of diamonds.

This idea has been successfully applied to define general sequent calculi (Seligman), natural deduction systems (Seligman, Brauner), and resolution calculi (Areces). It's also been applied with partial success to define display calculi (Demri and Goré).

Let's take a quick look at the way Torben Brauner handles natural deduction in hybrid logic.

Some basic natural deduction rules

$$\dfrac{\begin{matrix}[@_i\varphi]\\ \vdots \\ @_i\psi\end{matrix}}{@_i(\varphi \to \psi)}\ (\to I) \qquad \dfrac{@_i(\varphi \to \psi) \quad @_i\varphi}{@_i\psi}\ (\to E)$$

$$\dfrac{@_i\varphi}{@_k@_i\varphi}\ (@I) \qquad \dfrac{@_k@_i\varphi}{@_i\varphi}\ (@E)$$

Natural deduction rules for modalities

$$\dfrac{\begin{matrix}[@_i\Diamond j]\\ \vdots \\ @_j\varphi\end{matrix}}{@_i\Box\varphi}\ (\Box I)^* \qquad \dfrac{@_i\Box\varphi \quad @_i\Diamond k}{@_k\varphi}\ (\Box E)$$

$*$ j does not occur in $@_i\Box\varphi$ or in any undischarged assumptions other than the specified occurrences of $@_i\Diamond j$.

An example: $\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[@_i\Box(\varphi \to \psi)]^3 \quad [@_i\Diamond j]^1}{@_j(\varphi \to \psi)}\ (\Box E) \qquad \dfrac{[@_i\Box\varphi]^2 \quad [@_i\Diamond j]^1}{@_j\varphi}\ (\Box E)}{@_j\psi}\ (\to E)}{@_i\Box\psi}\ (\Box I)^1}{@_i(\Box\varphi \to \Box\psi)}\ (\to I)^2}{@_i(\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi))}\ (\to I)^3$$

Is any of this stuff implementable?

Yes — but we need to be careful. For example, the equality rules discussed today are nice for hand calculation, but naive computationally.

The **HTab** system (Areces and Hoffmann) implements more sophisticated rules (due to Bolander and Blackburn) which guarantee termination. The system is optimised in several ways, and although a recent system, is already a competitive prover.

And then there's resolution

A significant development is the adaptation of the resolution method for hybrid logic (Areces) and the implementation of the **HyLoRes** prover (Areces, Gorín, and Heguiabehere).

Strictly speaking, the method is resolution, plus a little paramodulation to handle the equality reasoning. The hybrid resolution rule is significantly simpler than other known approaches to modal resolution — @ and nominals allow us to pull resolvents out of the scope of modalities.

Many first-order resolution optimization techniques transfer to hybrid logic, and Areces and Gorin are currently incorporating such improvements into **HyLoRes**.

HTab and **HyLoRes** from the core of the new **InToHyLo** (inference Tools for Hybrid Logic System).

Summing up ...

- Orthodox modal logic demands proof methods that are applicable to a wide range to of logics. But because it is hard to extract information from under the scope of diamonds it has been forced to rely on Hilbert-systems, thereby sacrificing ease-of-use.
- The new tools offered by the basic hybrid language (nominals and @) enable us to define usable proof systems, such as tableau and natural deduction, basically because they make it easy to pull information out of modal scope.
- These proof methods can be generalized to a wide range of logics (completeness is automatic for pure formulas). Mature implementations now exist.

A home for modal logic

Claim: if you're doing traditional modal logic, you're working in the space carved out by hybrid logic with downarrow.

- We identify "locality" with "invariance under generated submodels."
- All traditional modal logics enjoy this property (though some newcomers, such as the global modality and the difference operator, explore what happens when you break locality).
- Hybrid logic with downarrow provides a comfortable home for traditional logics, performing such services as interpolation repair.

But we are getting *way* ahead of ourselves — let's first sit back and learn what downarrow actually does...

Example 1: Losers

In our first example, we'll think of the states of our models as people (so we're in a description logic style setting).

Example 1: Losers

In our first example, we'll think of the states of our models as people (so we're in a description logic style setting).

Suppose we define a loser to be someone who does not respect himself/herself Can we define this concept in the basic hybrid language?

Example 1: Losers

In our first example, we'll think of the states of our models as people (so we're in a description logic style setting).

Suppose we define a loser to be someone who does not respect himself/herself Can we define this concept in the basic hybrid language?

Well, we can get part of the way, for we can say things like:

$@_i \neg \langle \text{RESPECT} \rangle\, i$ *(i does not respect himself/herself)*

Example 1: Losers

In our first example, we'll think of the states of our models as people (so we're in a description logic style setting).

Suppose we define a loser to be someone who does not respect himself/herself Can we define this concept in the basic hybrid language?

Well, we can get part of the way, for we can say things like:

$@_i \neg \langle \text{RESPECT} \rangle\, i$ *(i does not respect himself/herself)*

$@_j \neg \langle \text{RESPECT} \rangle\, j$ *(j does not respect himself/herself)*

Example 1: Losers

In our first example, we'll think of the states of our models as people (so we're in a description logic style setting).

Suppose we define a loser to be someone who does not respect himself/herself Can we define this concept in the basic hybrid language?

Well, we can get part of the way, for we can say things like:

$@_i \neg \langle \text{RESPECT} \rangle\, i$ *(i does not respect himself/herself)*

$@_j \neg \langle \text{RESPECT} \rangle\, j$ *(j does not respect himself/herself)*

$@_k \neg \langle \text{RESPECT} \rangle\, k$ *(k does not respect himself/herself)*

Example 1: Losers

In our first example, we'll think of the states of our models as people (so we're in a description logic style setting).

Suppose we define a loser to be someone who does not respect himself/herself Can we define this concept in the basic hybrid language?

Well, we can get part of the way, for we can say things like:

$@_i \neg \langle \text{RESPECT} \rangle\, i$ *(i does not respect himself/herself)*

$@_j \neg \langle \text{RESPECT} \rangle\, j$ *(j does not respect himself/herself)*

$@_k \neg \langle \text{RESPECT} \rangle\, k$ *(k does not respect himself/herself)*

But none of these formulas pins down the concept of self-respect — only the concepts of self-respect for i, for j, for k, and so on. We need to abstract away from the effects of particular nominals (constants).

Losers via downarrow

With the aid of the downarrow operator, we can do precisely this:

$$\downarrow x.\neg \langle \text{RESPECT} \rangle\, x$$

This says: Let x be a temporary name for the point in the model at which the formula is being evaluated. Then x is not related to x by the RESPECT relation.

To put it another way, it says: this person x (whoever that is) does not respect himself/herself. In a sense, its what linguists call a deictic definition.

The formula is true of precisely those states of our models (people) who do not respect themselves, so we have defined the required concept.

Example 2: Locally reflected epistemic states

In our second example, we'll think of the states of our models as epistemic states, and the relation between states as meaning is an epistemic alternative to (so we're in a traditional agent-based setting).

Let's say that an epistemic state s is locally reflected if all epistemic alternatives t to s have s as an epistemic alternative.

More precisely, s is locally reflected iff $\forall t(sRt \rightarrow tRs)$. That is, s is a locally reflected state if it is symmetrically linked to other points in the model.

Is there a basic hybrid formula that (in any model) distinguishes locally reflected from non locally reflected states?

Well, we can try, but. . .

Well, we can try, but. . .

In particular, note that the formula $@_i\Box\Diamond i$ does not do what we want.

Well, we can try, but. . .

In particular, note that the formula $@_i\Box\Diamond i$ does not do what we want.

- In any particular model it merely asserts that the particular state named i is locally reflected ("symmetrically linked"). But that's not what we want.

Well, we can try, but. . .

In particular, note that the formula $@_i\Box\Diamond i$ does not do what we want.

- In any particular model it merely asserts that the particular state named i is locally reflected ("symmetrically linked"). But that's not what we want.
- On the other hand, if we insist that $@_i\Box\Diamond i$ is to be taken as a validity (that is, as an axiom) then we distinguish symmetric models from all other models. But that's not what we want either.

Well, we can try, but. . .

In particular, note that the formula $@_i\Box\Diamond i$ does not do what we want.

- In any particular model it merely asserts that the particular state named i is locally reflected ("symmetrically linked"). But that's not what we want.
- On the other hand, if we insist that $@_i\Box\Diamond i$ is to be taken as a validity (that is, as an axiom) then we distinguish symmetric models from all other models. But that's not what we want either.
- We want a formula that classifies the states of a model into locally reflected and non locally reflected states.

Locally reflected states via downarrow

We can do this with downarrow. Instead of $@_i\Box\Diamond i$ we use:

$$\downarrow x.\Box\Diamond x$$

Paraphrase this as follows: "this epistemic state x (whichever it might be) is such that all it's epistemic alternatives have x as an epistemic alternative."

Again, we're defining the required concept by some kind of deixis (this time, deictic reference to epistemic states, not people).

Technically, we bind a state variable x to the current state. (A state variable is just like a nominal, except that it can be bound, whereas ordinary nominals can't.)

Example 3: Problems and alarms

In this example we'll think of the states of our models as points of time (so we're in a temporal logic setting). The example is adapted from "Temporal Logic with Forgettable Past", Laroussinie, Markey, and Schnoebelen, 17th IEEE Symp. Logic in Computer Science (LICS 2002), Copenhagen, Denmark, July 2002.

Example 3: Problems and alarms

In this example we'll think of the states of our models as points of time (so we're in a temporal logic setting). The example is adapted from "Temporal Logic with Forgettable Past", Laroussinie, Markey, and Schnoebelen, 17th IEEE Symp. Logic in Computer Science (LICS 2002), Copenhagen, Denmark, July 2002.

- Suppose we are working with a system in which an alarm may go off (once) sometime in the future. We want to specify the following property: "Before the alarm goes off, there was a problem." Can we do this?

Example 3: Problems and alarms

In this example we'll think of the states of our models as points of time (so we're in a temporal logic setting). The example is adapted from "Temporal Logic with Forgettable Past", Laroussinie, Markey, and Schnoebelen, 17th IEEE Symp. Logic in Computer Science (LICS 2002), Copenhagen, Denmark, July 2002.

- Suppose we are working with a system in which an alarm may go off (once) sometime in the future. We want to specify the following property: "Before the alarm goes off, there was a problem." Can we do this?
- Easy: $[\mathrm{F}]\,(\mathrm{alarm} \to \langle \mathrm{P} \rangle\, \mathrm{problem})$ specifies this. We don't even need hybrid logic.

Resetting the alarm

Now suppose that the alarm has a reset button, and we want to state that the previous specification holds after any reset. Can we say this?

Resetting the alarm

Now suppose that the alarm has a reset button, and we want to state that the previous specification holds after any reset. Can we say this?

- Here's an attempt: $[\mathrm{F}]\,(\mathrm{reset} \to [\mathrm{F}]\,(\mathrm{alarm} \to \langle \mathrm{P} \rangle\, \mathrm{problem}))$

Resetting the alarm

Now suppose that the alarm has a reset button, and we want to state that the previous specification holds after any reset. Can we say this?

- Here's an attempt: $[\mathrm{F}]\,(\mathrm{reset} \to [\mathrm{F}]\,(\mathrm{alarm} \to \langle \mathrm{P} \rangle\, \mathrm{problem}))$
- But this is probably not what we want — a problem that occurred before the reset may account for the alarm going off.

Resetting the alarm with downarrow

But with the aid of $\downarrow$ we can specify what we want. We dynamically name the spot where the reset occurred by binding the state variable x to it, and then demand that the problem occurred later than this:

$$[\mathrm{F}]\,(\mathrm{reset} \to {\downarrow} x.[\mathrm{F}]\,(\mathrm{alarm} \to \langle \mathrm{P} \rangle\,(\mathrm{problem} \wedge \langle \mathrm{P} \rangle\, x)))$$

Example 4: The *Until* operator

In this example we'll continue to think of the states of our models as points of time (so we're still doing temporal logic).

Hans Kamp's celebrated *Until* operator is a binary modality with the following satisfaction definition:

$$\mathcal{M}, w \Vdash \mathit{Until}(\varphi, \psi) \quad \text{iff} \quad \exists v(w < v \ \&\ \mathcal{M}, v \Vdash \varphi \ \& \ \forall u(w < u < v \Rightarrow \mathcal{M}, u \Vdash \psi))$$

This operator (and some of its variants) has proved extremely useful as a specification tool. Can we define it in basic hybrid logic?

Defining *Until* with downarrow

Defining *Until* with downarrow

No, we can't. But we can with the help of downarrow:

$$Until(\varphi, \psi) \ := \ \downarrow x.\Diamond\downarrow y.(\varphi \wedge @_x\Box(\Diamond y \rightarrow \psi)).$$

This says: name the present state x. Then, by looking forward we can see a state (which we label y) such that φ is true at y, and every state between x and y verifies ψ.

Note the use of $@_x$ to jump back to x, the starting point. This is the first glimpse of a theme that echos through today's lecture: $\downarrow$ and @ work well together. We can use $\downarrow$ to 'store' some state of interest, and @ to 'retrieve' it when needed.

Syntax

- Here are the required syntactic changes. Choose a denumerably infinite set SVAR $= \{x, y, z \dots\}$, the set of state variables, disjoint from PROP, NOM and MOD.
- Like nominals, state variables are atomic formulas which name states, but unlike nominals they can be bound.
- The hybrid language with downarrow (over PROP, NOM, MOD and SVAR) is defined as follows:

$$\begin{array}{lll} \text{WFF} & := & x \mid i \mid \text{p} \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \\ & & \mid \varphi \rightarrow \psi \mid \langle \text{M} \rangle \varphi \mid [\text{M}] \varphi \mid @_i\varphi \mid @_x\varphi \mid \downarrow x.\varphi \end{array}$$

- Free and bound occurrences of state variables are defined in the expected way, with $\downarrow$ as the only binder. A sentence is a formula containing no free state variables.

Semantics

- Models $\mathcal{M}$ for hybrid languages with downarrow are just the hybrid models we are used to (as usual, nominals are assigned singletons).
- Given a model $\mathcal{M} = (W, R, V)$, an assignment on $\mathcal{M}$ is a function $g : \text{SVAR} \longrightarrow W$. (Thus an assignment makes a state variable true at precisely one state.)
- Assignments will be used to interpret free state variables Tarski-style. We merely relativise the clauses of the satisfaction definition for the basic hybrid language to assignments, and add the three new clauses we require. Here's how ...

Satisfaction Definition

$$\begin{array}{lll} \mathcal{M}, g, w \Vdash x & \text{iff} & w = g(x) \text{ where } x \in \text{SVAR} \\ \mathcal{M}, g, w \Vdash @_x\varphi & \text{iff} & \mathcal{M}, g, g(x) \Vdash \varphi \\ \mathcal{M}, g, w \Vdash \varphi \wedge \psi & \text{iff} & \mathcal{M}, g, w \Vdash \varphi \text{ and } \mathcal{M}, g, w \Vdash \psi \\ \mathcal{M}, g, w \Vdash \Diamond\varphi & \text{iff} & \exists w'(wRw' \ \& \ \mathcal{M}, g, w' \Vdash \varphi) \\ \mathcal{M}, g, w \Vdash \downarrow x.\varphi & \text{iff} & \mathcal{M}, g', w \Vdash \varphi, \text{ where } g' \stackrel{x}{\sim} g \text{ and } g'(x) = w \end{array}$$

The fifth clause defines $\downarrow$ to be an operator that binds variables to the state w at which evaluation is being performed. The notation $g' \stackrel{x}{\sim} g$ means that g' is the assignment that differs from g, if at all, only in what it assigns to x. By stipulating that $g'(x)$ is to be w, we bind a label to the here-and-now.

For sentences φ, we can simply write $\mathcal{M}, w \Vdash \downarrow x.\varphi$ — why is this?

Standard Translation

Assume we're using the same symbols for both state variables and first-order variables. Let s be a metavariable over state variables and nominals.

$$\begin{array}{lll} \text{ST}_x(y) & = & (y = x) \\ \text{ST}_x(@_s\varphi) & = & \text{ST}_s(\varphi) \\ \text{ST}_x(\downarrow y.\varphi) & = & \exists y(x = y \wedge \text{ST}_x(\varphi)) \end{array}$$

This translation is satisfaction preserving, so hybrid logic with downarrow is a fragment of the correspondence language (with constants and equalities). We'll see later which fragment it corresponds to.

Tableau rules

We only need to make two changes. First, we need to let our previous tableau rules apply when the subscript on @ is a state variable rather than a nominal.

Second, we add the following two rules to cope with $\downarrow$. In the following rule, s is used as a metavariable over nominals and state variables:

$$\frac{@_s\downarrow x.\varphi}{@_s\varphi[x \leftarrow s]} \qquad \frac{\neg @_s\downarrow x.\varphi}{\neg @_s\varphi[x \leftarrow s]}$$

If s is a variable, before substituting we rename bound occurrences of s in φ to prevent accidental capture.

Example: $\downarrow x.x$

Example: $\downarrow x.x$

1 $\neg @_i\downarrow x.x$

Example: $\downarrow x.x$

1	$\neg @_i \downarrow x.x$	
2	$\neg @_i i$	$\neg\downarrow$ rule on 1

Example: $\downarrow x.x$

1	$\neg @_i \downarrow x.x$	
2	$\neg @_i i$	$\neg\downarrow$ rule on 1
3	$@_i i$	Ref

Example: $\downarrow x.x$

1	$\neg @_i \downarrow x.x$	
2	$\neg @_i i$	$\neg\downarrow$ rule on 1
3	$@_i i$	Ref
	$\bot_{2,3}$	

Example: $\downarrow x.\varphi \leftrightarrow \neg\downarrow x.\neg\varphi$

Example: $\downarrow x.\varphi \leftrightarrow \neg\downarrow x.\neg\varphi$

That is, like @, the downarrow binder is self dual. Let's prove the left-to right direction of this equivalence:

1	$\neg @_i(\downarrow x.\varphi \rightarrow \neg\downarrow x.\neg\varphi)$	

Example: $\downarrow x.\varphi \leftrightarrow \neg\downarrow x.\neg\varphi$

That is, like @, the downarrow binder is self dual. Let's prove the left-to right direction of this equivalence:

1	$\neg @_i(\downarrow x.\varphi \rightarrow \neg\downarrow x.\neg\varphi)$	
2	$@_i \downarrow x.\varphi$	
$2'$	$\neg @_i \neg\downarrow x.\neg\varphi$	Propositional rule on 1

Example: $\downarrow x.\varphi \leftrightarrow \neg\downarrow x.\neg\varphi$

That is, like @, the downarrow binder is self dual. Let's prove the left-to right direction of this equivalence:

1	$\neg @_i(\downarrow x.\varphi \rightarrow \neg\downarrow x.\neg\varphi)$	
2	$@_i \downarrow x.\varphi$	
$2'$	$\neg @_i \neg\downarrow x.\neg\varphi$	Propositional rule on 1
3	$@_i \downarrow x.\neg\varphi$	Propositional rule on $2'$

Example: $\downarrow x.\varphi \leftrightarrow \neg\downarrow x.\neg\varphi$

That is, like @, the downarrow binder is self dual. Let's prove the left-to right direction of this equivalence:

1	$\neg @_i(\downarrow x.\varphi \rightarrow \neg\downarrow x.\neg\varphi)$	
2	$@_i \downarrow x.\varphi$	
$2'$	$\neg @_i \neg\downarrow x.\neg\varphi$	Propositional rule on 1
3	$@_i \downarrow x.\neg\varphi$	Propositional rule on $2'$
4	$@_i \neg\varphi[x \leftarrow i]$	$\downarrow$ rule on 3

Example: $\downarrow x.\varphi \leftrightarrow \neg\downarrow x.\neg\varphi$

That is, like @, the downarrow binder is self dual. Let's prove the left-to right direction of this equivalence:

1	$\neg @_i(\downarrow x.\varphi \rightarrow \neg\downarrow x.\neg\varphi)$	
2	$@_i\downarrow x.\varphi$	
2′	$\neg @_i\neg\downarrow x.\neg\varphi$	Propositional rule on 1
3	$@_i\downarrow x.\neg\varphi$	Propositional rule on 2′
4	$@_i\neg\varphi[x \leftarrow i]$	↓ rule on 3
5	$\neg @_i\varphi[x \leftarrow i]$	Propositional rule on 4

Example: $\downarrow x.\varphi \leftrightarrow \neg\downarrow x.\neg\varphi$

That is, like @, the downarrow binder is self dual. Let's prove the left-to right direction of this equivalence:

1	$\neg @_i(\downarrow x.\varphi \rightarrow \neg\downarrow x.\neg\varphi)$	
2	$@_i\downarrow x.\varphi$	
2′	$\neg @_i\neg\downarrow x.\neg\varphi$	Propositional rule on 1
3	$@_i\downarrow x.\neg\varphi$	Propositional rule on 2′
4	$@_i\neg\varphi[x \leftarrow i]$	↓ rule on 3
5	$\neg @_i\varphi[x \leftarrow i]$	Propositional rule on 4
6	$@_i\varphi[x \leftarrow i]$	↓ rule on 2

Example: $\downarrow x.\varphi \leftrightarrow \neg\downarrow x.\neg\varphi$

That is, like @, the downarrow binder is self dual. Let's prove the left-to right direction of this equivalence:

1	$\neg @_i(\downarrow x.\varphi \rightarrow \neg\downarrow x.\neg\varphi)$	
2	$@_i\downarrow x.\varphi$	
2′	$\neg @_i\neg\downarrow x.\neg\varphi$	Propositional rule on 1
3	$@_i\downarrow x.\neg\varphi$	Propositional rule on 2′
4	$@_i\neg\varphi[x \leftarrow i]$	↓ rule on 3
5	$\neg @_i\varphi[x \leftarrow i]$	Propositional rule on 4
6	$@_i\varphi[x \leftarrow i]$	↓ rule on 2
	$\bot_{5,6}$	

Completeness

This tableau system is (sound and) complete with respect to the class of all models.

Nonetheless, as was explained in yesterday's lecture, we are often interested in deduction over other classes of models. Can the tableau system be extended to deal with reasoning over other classes of models?

Yes, it can — and once again it's pure formulas that make things easy.

Pure formulas

- As before, a pure formula is simply a formula not containing any propositional symbols.
- But this means that pure formulas may contain state variables and ↓, not just nominals, ⊥ and ⊤, so we can define a lot more frame classes than before.
- Nonetheless, completeness is still automatic. Recall that if $@_i\varphi$ be a pure formula, whose nominals (if any) are $i, i_1, \ldots, i_n$, then we can turn it into the following tableau rule:

$$\frac{(j, j_1, \ldots, j_n \text{ on branch})}{@_i\varphi[i \leftarrow j, i_1 \leftarrow j_1, \ldots, i_n \leftarrow j_n]}$$

Frame definability and deduction match for pure formulas

Completeness Theorem Suppose you extend the basic tableau system with the tableau rules for the pure formulas $@_j\varphi$, ..., $@_k\psi$ (that is, the rules of the form just described). Then the resulting system is (sound and) complete with respect to the class of frames defined by these formulas.

That is, the frame-defining and deductive powers of pure formulas match perfectly — even when ↓ has been added to the language.

Towards the logic of locality

Towards the logic of locality

- But now for the fundamental question: what exactly is hybrid logic with downarrow?

Towards the logic of locality

- But now for the fundamental question: what exactly is hybrid logic with downarrow?
- We know what basic modal logic is: it's the bisimulation invariant fragment of first-order logic.

Towards the logic of locality

- But now for the fundamental question: what exactly is hybrid logic with downarrow?
- We know what basic modal logic is: it's the bisimulation invariant fragment of first-order logic.

- We know what basic hybrid logic is: it's the bisimulation-with-constants invariant fragment of first-order logic.

Towards the logic of locality

- But now for the fundamental question: what exactly is hybrid logic with downarrow?
- We know what basic modal logic is: it's the bisimulation invariant fragment of first-order logic.
- We know what basic hybrid logic is: it's the bisimulation-with-constants invariant fragment of first-order logic.

- As we shall learn, hybrid logic with downarrow also corresponds to a neat fragment of first-order logic: it's the first-order logic of locality.

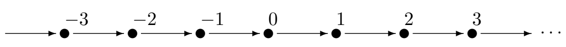
Towards the logic of locality

- But now for the fundamental question: what exactly is hybrid logic with downarrow?
- We know what basic modal logic is: it's the bisimulation invariant fragment of first-order logic.
- We know what basic hybrid logic is: it's the bisimulation-with-constants invariant fragment of first-order logic.
- As we shall learn, hybrid logic with downarrow also corresponds to a neat fragment of first-order logic: it's the first-order logic of locality.

To understand what this means we're going to need to learn something about submodels and generated submodels. . .

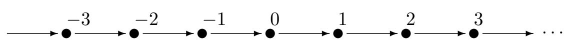
Submodels

Suppose $\mathcal{M}$ is a model based on this frame (the integers in their usual order):

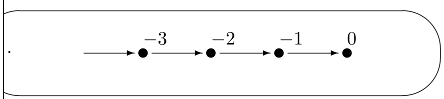


Submodels

Suppose $\mathcal{M}$ is a model based on this frame (the integers in their usual order):

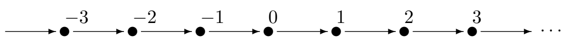


Suppose we form a submodel $\mathcal{M}^-$ of $\mathcal{M}$ by throwing away all the positive numbers, and restricting the original valuation (whatever it was) to the remaining numbers:

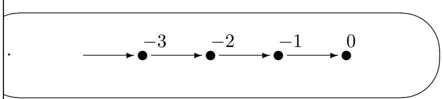


Submodels

Suppose $\mathcal{M}$ is a model based on this frame (the integers in their usual order):



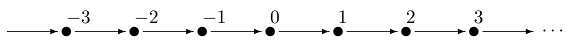
Suppose we form a submodel $\mathcal{M}^-$ of $\mathcal{M}$ by throwing away all the positive numbers, and restricting the original valuation (whatever it was) to the remaining numbers:



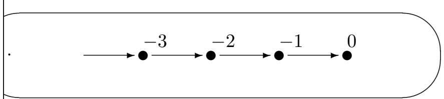
Can an orthodox modal language detect the difference between the two model?

Submodels

Suppose $\mathcal{M}$ is a model based on this frame (the integers in their usual order):



Suppose we form a submodel $\mathcal{M}^-$ of $\mathcal{M}$ by throwing away all the positive numbers, and restricting the original valuation (whatever it was) to the remaining numbers:

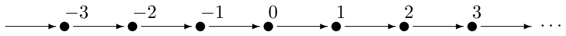


Can an orthodox modal language detect the difference between the two model?

Yes! $\mathcal{M}, 0 \Vdash \Diamond\top$, but $\mathcal{M}^-, 0 \nVdash \Diamond\top$

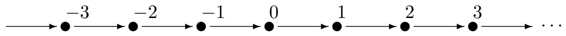
Another submodel

Again $\mathcal{M}$ is a model based on the integers in their usual order:

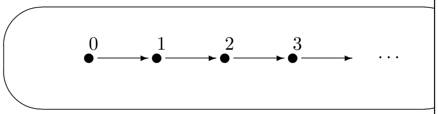


Another submodel

Again $\mathcal{M}$ is a model based on the integers in their usual order:

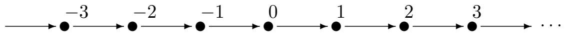


This time, suppose we form a submodel of $\mathcal{M}^+$ of $\mathcal{M}$ obtained by throwing away the negative numbers, and restricting the original valuation to what remains:

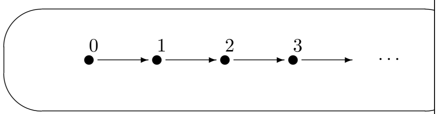


Another submodel

Again $\mathcal{M}$ is a model based on the integers in their usual order:



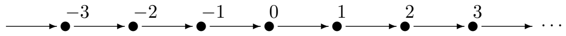
This time, suppose we form a submodel of $\mathcal{M}^+$ of $\mathcal{M}$ obtained by throwing away the negative numbers, and restricting the original valuation to what remains:



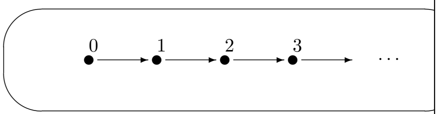
Can an orthodox modal language detect the difference between the two model?

Another submodel

Again $\mathcal{M}$ is a model based on the integers in their usual order:



This time, suppose we form a submodel of $\mathcal{M}^+$ of $\mathcal{M}$ obtained by throwing away the negative numbers, and restricting the original valuation to what remains:



Can an orthodox modal language detect the difference between the two model?

No! The two models make the exactly the same formulas true.

Why the difference?

Why the difference?

- Well, in the second example the two models were bisimilar, and in the first example they weren't.

Why the difference?

- Well, in the second example the two models were bisimilar, and in the first example they weren't.
- But there's a more direct intuition: the second model consisted of the point 0 and all it's successors (that is, it's the submodel generated by the point 0).

Why the difference?

- Well, in the second example the two models were bisimilar, and in the first example they weren't.
- But there's a more direct intuition: the second model consisted of the point 0 and all it's successors (that is, it's the submodel generated by the point 0).
- To put it another way, point generation selects all the points that are reachable from the evaluation state by chaining through the relation(s). It selects precisely the points needed to satisfy a formula at some particular location, and ignores the rest.

Point generated submodels

Point generated submodels Let $\mathcal{M} = (W, R, V)$ be a model, and $w \in W$. Let $W_w = \{w' \in W \mid wR^*w'\}$, where R^* is the reflexive transitive closure of R. Then $\mathcal{M}_w$, the submodel of $\mathcal{M}$ generated by w is the model (W_w, R_w, V_w) where R_w and V_w are the restrictions of R and V, respectively, to W_w.

Point generated submodels

Point generated submodels Let $\mathcal{M} = (W, R, V)$ be a model, and $w \in W$. Let $W_w = \{w' \in W \mid wR^*w'\}$, where R^* is the reflexive transitive closure of R. Then $\mathcal{M}_w$, the submodel of $\mathcal{M}$ generated by w is the model (W_w, R_w, V_w) where R_w and V_w are the restrictions of R and V, respectively, to W_w.

Proposition: Let $\mathcal{M}$ be a model and $\mathcal{M}_w$ any of its point generated submodels. Then for any orthodox modal formula φ, and any point u in $\mathcal{M}_w$ we have

$$\mathcal{M}, u \Vdash \varphi \text{ iff } \mathcal{M}_w, u \Vdash \varphi$$

In words: model satisfaction is invariant for point generated submodels.

Proof: By direct induction on the structure of φ, or by observing that point generation always results in bisimilar models.

Does this invariance hold for all hybrid formulas?

Does this invariance hold for all hybrid formulas?

No!

Does this invariance hold for all hybrid formulas?

No!

Why not?

Does this invariance hold for all hybrid formulas?

No!

Why not?

Because nominals and free variables may denote non-local points — that is, points that do not belong to the generated submodel. And then we can jump non-locally using @.

Does this invariance hold for all hybrid formulas?

No!

Why not?

Because nominals and free variables may denote non-local points — that is, points that do not belong to the generated submodel. And then we can jump non-locally using @.

Let's restrict our attention to nominal-free sentences. All occurrences of @ in such formulas are bound by $\downarrow$ — surely this can only lead to "local jumping"?

Does this invariance hold for all hybrid formulas?

No!

Why not?

Because nominals and free variables may denote non-local points — that is, points that do not belong to the generated submodel. And then we can jump non-locally using @.

Let's restrict our attention to nominal-free sentences. All occurrences of @ in such formulas are bound by $\downarrow$ — surely this can only lead to "local jumping"?

This idea is correct. How do we prove it?

Nominal-free sentences are invariant under generated submodels

Lemma: $\mathcal{M}$ be a model, let $\mathcal{M}_w$ be any of its point generated submodels, and g be an assignment sending all state variables to points in $\mathcal{M}_w$. Then for any nominal-free formula φ (in the hybrid language with $\downarrow$) and any point u in $\mathcal{M}_w$

$$\mathcal{M}, u, g \Vdash \varphi \text{ iff } \mathcal{M}_w, u, g \Vdash \varphi$$

Nominal-free sentences are invariant under generated submodels

Lemma: $\mathcal{M}$ be a model, let $\mathcal{M}_w$ be any of its point generated submodels, and g be an assignment sending all state variables to points in $\mathcal{M}_w$. Then for any nominal-free formula φ (in the hybrid language with $\downarrow$) and any point u in $\mathcal{M}_w$

$$\mathcal{M}, u, g \Vdash \varphi \text{ iff } \mathcal{M}_w, u, g \Vdash \varphi$$

Proof: By induction on the structure of φ. In the step for subformulas of the form $\downarrow y.\psi$ observe that y is assigned a value in $\mathcal{M}_w$, hence the variant assignment g' satisfies the inductive hypothesis.

Nominal-free sentences are invariant under generated submodels

Lemma: $\mathcal{M}$ be a model, let $\mathcal{M}_w$ be any of its point generated submodels, and g be an assignment sending all state variables to points in $\mathcal{M}_w$. Then for any nominal-free formula φ (in the hybrid language with $\downarrow$) and any point u in $\mathcal{M}_w$

$$\mathcal{M}, u, g \Vdash \varphi \text{ iff } \mathcal{M}_w, u, g \Vdash \varphi$$

Proof: By induction on the structure of φ. In the step for subformulas of the form $\downarrow y.\psi$ observe that y is assigned a value in $\mathcal{M}_w$, hence the variant assignment g' satisfies the inductive hypothesis.

Corollary: The truth of pure nominal free sentences is invariant under generated submodels.

What about first-order formulas?

- A first-order formula in one free variable $\varphi(x)$ is invariant under point generated submodels if for any model $\mathcal{M}$, any of its point generated submodels $\mathcal{M}_w$, and any point u in $\mathcal{M}_w$, $\mathcal{M} \models \varphi[u]$ iff $\mathcal{M}' \models \varphi[u]$.
- Obviously not all first-order formulas are invariant under generated submodels — first-order logic is clearly non-local!
- But some are. Which ones? That is, what is the first-order logic of locality?

The logic of locality

Theorem: A first-order formula in one free variable is invariant for generated submodels iff it is equivalent to the standard translation of a nominal-free sentence (of the hybrid language with downarrow).

That is, hybrid logic with downarrow is precisely the first-order logic of locality.

For the original proof see "Hybrid Logics: Characterization, Interpolation and Complexity", Areces, Blackburn and Marx, *Journal of Symbolic Logic*, 66:977-1009, 2001.

For an even better proof see Balder ten Cate's 2004 Amsterdam PhD thesis, *Model Theory for Extended Modal Languages*. (We may discuss this proof on Friday.)

Interpolation

A logic has the interpolation property if whenever

$$\models \varphi \to \psi$$

then there is some formula θ containing only non-logical symbols common to φ and ψ such that:

$$\models \varphi \to \theta \quad \text{and} \quad \models \theta \to \psi.$$

Roughly speaking, if a logic enjoys interpolation, then validity can always be 'filtered through' the common information bearing elements of the language.

Interpolation in modal logic

Interpolation in modal logic

- Orthodox propositional modal logic is not particularly well behaved with respect to interpolation.

Interpolation in modal logic

- Orthodox propositional modal logic is not particularly well behaved with respect to interpolation.
- And neither is basic hybrid logic: we'll now see that interpolation fails in the basic hybrid language.

Interpolation in modal logic

- Orthodox propositional modal logic is not particularly well behaved with respect to interpolation.
- And neither is basic hybrid logic: we'll now see that interpolation fails in the basic hybrid language.
- However we'll immediately be able to 'repair' this failure with $\downarrow$. And in fact, $\downarrow$ can repair systematically repair interpolation failures.

Interpolation failure in basic hybrid logic

Interpolation failure in basic hybrid logic

In Lecture 1 we gave a tableau proof of

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

Hence this formula is valid. So if the basic hybrid language enjoys interpolation then there should exist an interpolating θ such that

$$\models (\Diamond \mathrm{p} \wedge \Diamond \neg \mathrm{p}) \rightarrow \theta \quad \text{and} \quad \theta \rightarrow (\Box(\mathrm{q} \rightarrow i) \rightarrow \Diamond \neg \mathrm{q}).$$

Note that θ must be in the empty language (that is, it must be built up solely from $\top$ and $\bot$) as $\{p\} \cap \{i, q\} = \emptyset$.

$\models (\Diamond \mathrm{p} \wedge \Diamond \neg \mathrm{p}) \rightarrow \theta$ and $\models \theta \rightarrow (\Box(\mathrm{q} \rightarrow i) \rightarrow \Diamond \neg \mathrm{q})$

$\models (\Diamond \mathrm{p} \wedge \Diamond \neg \mathrm{p}) \rightarrow \theta$ and $\models \theta \rightarrow (\Box(\mathrm{q} \rightarrow i) \rightarrow \Diamond \neg \mathrm{q})$

- What would an interpolant look like? Well, a θ saying "I have at least two successors" (in the empty language) would do.

$\models (\Diamond \mathrm{p} \wedge \Diamond \neg \mathrm{p}) \rightarrow \theta$ and $\models \theta \rightarrow (\Box(\mathrm{q} \rightarrow i) \rightarrow \Diamond \neg \mathrm{q})$

- What would an interpolant look like? Well, a θ saying "I have at least two successors" (in the empty language) would do.
- Now, $\Box\bot$ says "I have zero successors" (in the empty language).

$\models (\Diamond \mathrm{p} \wedge \Diamond \neg \mathrm{p}) \rightarrow \theta$ and $\models \theta \rightarrow (\Box(\mathrm{q} \rightarrow i) \rightarrow \Diamond \neg \mathrm{q})$

- What would an interpolant look like? Well, a θ saying "I have at least two successors" (in the empty language) would do.
- Now, $\Box\bot$ says "I have zero successors" (in the empty language).
- And $\Diamond\top$ says "I have at least one successor" (in the empty language).

$\models (\Diamond p \wedge \Diamond \neg p) \rightarrow \theta$ and $\models \theta \rightarrow (\Box(q \rightarrow i) \rightarrow \Diamond \neg q)$

- What would an interpolant look like? Well, a θ saying "I have at least two successors" (in the empty language) would do.
- Now, $\Box\bot$ says "I have zero successors" (in the empty language).
- And $\Diamond\top$ says "I have at least one successor" (in the empty language).
- But it seems impossible to express "I have at least two successors" (in the empty language). And there doesn't seem to be any other candidate.

$\models (\Diamond p \wedge \Diamond \neg p) \rightarrow \theta$ and $\models \theta \rightarrow (\Box(q \rightarrow i) \rightarrow \Diamond \neg q)$

- What would an interpolant look like? Well, a θ saying "I have at least two successors" (in the empty language) would do.
- Now, $\Box\bot$ says "I have zero successors" (in the empty language).
- And $\Diamond\top$ says "I have at least one successor" (in the empty language).
- But it seems impossible to express "I have at least two successors" (in the empty language). And there doesn't seem to be any other candidate.
- And a simple bisimulation argument shows that no interpolant exists.

But what if we also had $\downarrow$ at our disposal?

But what if we also had $\downarrow$ at our disposal?

- The pure, nominal-free, sentence $\downarrow x.\Diamond\downarrow y.@_x\Diamond\neg y$ says that there are at least two distinct accessible states.
- Intuitively, because $\downarrow$ binds variables, we can say a lot (even in the empty language).
- This suggests that although interpolation fails for the basic hybrid language, it might holds for the richer language containing $\downarrow$. And in fact this is just the way things work out...

Hybrid logic with $\downarrow$ has interpolation

Theorem: Suppose we are working with the hybrid language with $\downarrow$. Then the logic of any class of frames definable by a pure, nominal-free, sentence of this language enjoys interpolation.

Proof:
For a model-theoretic proof (using a Chang and Keisler style construction) see "Hybrid Logics: Characterization, Interpolation and Complexity", Areces, Blackburn and Marx, *Journal of Symbolic Logic*, 66:977-1009, 2001. (You will probably see this proof on Friday.)

For a constructive proof (using tableau) see "Constructive interpolants for every bounded fragment definable hybrid logic", Blackburn and Marx, *Journal of Symbolic Logic*, 68(2), 463-480, 2003.

The finite model property

- A language has the finite model property if any satisfiable formula in the language can be satisfied on a finite model.
- The orthodox propositional modal language has the finite model property, and so does the basic hybrid language.
- Viewed negatively, this means that these languages are too weak to define infinite structures.
- Viewed positively, it means that we never need to bother about with infinite structures when working with these languages.

First-order logic lacks the finite model property

Consider the following first-order formulas:

- $\forall x \neg R(x, x)$ (Irreflexivity)
- $\forall x \exists y R(x, y)$ (Unboundedness)
- $\forall x \forall y (R(x, y) \wedge R(y, z) \rightarrow R(x, z))$ (Transitivity)

Any model for these formulas (for example, the natural numbers under their usual ordering) is called a unbounded strict total order. It is not hard to see that any unbounded strict total order is infinite. So first-order logic lack the finite model property.

Hybrid logic with $\downarrow$ also lacks the finite model property

- More difficult to prove, for we lack the globality of first-order logic.
- However we can show this using a spypoint argument.
- We shall define a certain sentence and show that all models satisfying it contain a point s (the spypoint) that can see strict unbounded total order.

A spypoint argument

Consider what any model of the following formula must contain:

$$
\begin{aligned}
& @_s\Box\Box\downarrow x.@_s\Diamond x \\
\wedge\ \ & @_s\Diamond\neg s \\
\wedge\ \ & @_s\Box\Diamond\top \\
\wedge\ \ & @_s\Box\downarrow x.\neg\Diamond x \\
\wedge\ \ & @_s\Box\downarrow x.\Box\Box\downarrow y.@_x\Diamond y
\end{aligned}
$$

This formula has some obvious models.

Moreover, any model for this formula must contain a point s such that the set of points B that s is related to is an unbounded strict total order — and hence infinite.

Hybrid logic with ↓ is undecidable

- We have stepped over an important boundary: adding ↓ has cost us decidability.
- In fact, even the fragment consisting of pure, nominal-free, @-free sentences is undecidable.
- This can also be proved using a spypoint argument. Basic technique is to use the spypoint as a vantage point surveying a coding of an undecidable problem. (See "Hybrid Logics: Characterization, Interpolation and Complexity", Areces, Blackburn and Marx, *Journal of Symbolic Logic*, 66:977-1009, 2001.)
- You'll probably see this in Thursday's lecture ...

Two comments on undecidability

- One interesting decidable fragment is known. Maarten Marx has shown that the fragment in which □ never occurs under the scope of ↓ is decidable (and in fact EXPTIME-complete). This fragment can handle some useful description logic definitions.
- Because downarrow binding is local, we always know which substitutions we have to perform. There is no need for Skolem functions or unification. It may be that theorem provers will perform well on "typical" formulas. The **HyLoRes** prover handles downarrow, and it is hoped to optimize it's performance for this binder.

Summing up ...

- We motivated the idea of binding variables to states locally, and introduced ↓ which lets us dynamically name the here-and-now.
- By doing this we have captured precisely the first-order logic of locality. Completeness and interpolation results hold for all local logics. Although local, the existence of infinite models can be forced, and the system is undecidable.