Expressivity and Inference in Hybrid Logic

Patrick Blackburn

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

The Second Tsinghua Logic Summer School

June 27 – July 3, 2022, Beijing, China

What we did yesterday...

- I told you about $\mathbf{K}_h$ system — a sound and complete axiom system for minimal hybrid logic.
- I explained why $\mathbf{K}_h$ was not the axiomatization we wanted.
- I introduced the NAME and PASTE rules rules, and showed how to prove a Hybrid Lindenbaum Lemma.
- This led to a completeness result (using Henkin's second idea) for the $\mathbf{K}_h + \text{RULES}$ axiom system, with respect to named models.
- This completeness result was for the minimal logic — and any other logic obtained by adding pure axioms.
- I tried to make the link between the NAME and PASTE rules and hybrid tableaus clear.
- I said that hybrid logics with pure axioms are like first-order theories.

MCSs within an MCS

MCSs within an MCS

- I told you about MCSs hidden inside other MCSs; I want to make this more intuitive.

MCSs within an MCS

- I told you about MCSs hidden inside other MCSs; I want to make this more intuitive.
- Suppose we are in world described by $\Sigma = \{p, k, \Diamond i, @_i \neg p, \Diamond j, @_i j \ldots\}$.

MCSs within an MCS

- I told you about MCSs hidden inside other MCSs; I want to make this more intuitive.
- Suppose we are in world described by $\Sigma = \{p, k, \Diamond i, @_i \neg p, \Diamond j, @_i j \ldots\}$.
- As it describes a world, Σ, must be an MCS. It must describe everything.

MCSs within an MCS

- I told you about MCSs hidden inside other MCSs; I want to make this more intuitive.
- Suppose we are in world described by $\Sigma = \{p, k, \Diamond i, @_i \neg p, \Diamond j, @_i j \ldots\}$.
- As it describes a world, Σ, must be an MCS. It must describe everything.
- So Σ must contain:

MCSs within an MCS

- I told you about MCSs hidden inside other MCSs; I want to make this more intuitive.
- Suppose we are in world described by $\Sigma = \{p, k, \Diamond i, @_i \neg p, \Diamond j, @_i j \ldots\}$.
- As it describes a world, Σ, must be an MCS. It must describe everything.
- So Σ must contain:
  - Complete information about world i. This will all the formulas of the form $@_i \varphi$. So all these formulas φ form an MCS.

MCSs within an MCS

- I told you about MCSs hidden inside other MCSs; I want to make this more intuitive.
- Suppose we are in world described by $\Sigma = \{p, k, \Diamond i, @_i \neg p, \Diamond j, @_i j \ldots\}$.
- As it describes a world, Σ, must be an MCS. It must describe everything.
- So Σ must contain:
  - Complete information about world i. This will all the formulas of the form $@_i \varphi$. So all these formulas φ form an MCS.
  - Complete information about world j. This will all the formulas of the form $@_j \varphi$. So all these formulas φ form an MCS too. And it will be identical to the i-MCS.

MCSs within an MCS

- I told you about MCSs hidden inside other MCSs; I want to make this more intuitive.
- Suppose we are in world described by $\Sigma = \{p, k, \Diamond i, @_i \neg p, \Diamond j, @_i j \ldots\}$.
- As it describes a world, Σ, must be an MCS. It must describe everything.
- So Σ must contain:
  - Complete information about world i. This will all the formulas of the form $@_i\varphi$. So all these formulas φ form an MCS.
  - Complete information about world j. This will all the formulas of the form $@_j\varphi$. So all these formulas φ form an MCS too. And it will be identical to the i-MCS.
  - Complete information about itself, world k. This will all the formulas of the form $@_k\varphi$. So all these formulas φ form an MCS too.

MCSs within an MCS

- I told you about MCSs hidden inside other MCSs; I want to make this more intuitive.
- Suppose we are in world described by $\Sigma = \{p, k, \Diamond i, @_i \neg p, \Diamond j, @_i j \ldots\}$.
- As it describes a world, Σ, must be an MCS. It must describe everything.
- So Σ must contain:
  - Complete information about world i. This will all the formulas of the form $@_i\varphi$. So all these formulas φ form an MCS.
  - Complete information about world j. This will all the formulas of the form $@_j\varphi$. So all these formulas φ form an MCS too. And it will be identical to the i-MCS.
  - Complete information about itself, world k. This will all the formulas of the form $@_k\varphi$. So all these formulas φ form an MCS too.
- It all comes down to Robinson Diagrams.

Today: Adding Downarrow

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

- Motivating downarrow.
- Standard translation, tableau and axiomatics.
- What is hybrid logic enriched with downarrow?
- Interpolation.
- Finite model property and decidability.
- Technical themes: interaction between $\downarrow$ and @, pure sentences, point generated submodels, interpolation, and spypoint arguments.
- Conceptual theme: Downarrow is the logic of locality.

In today's **tutorial session** **(1×45 minutes)** we can have our usual Q&A or maybe look at answers to Homework 3.

Example 1: Confidence

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: Confidence

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 confident person to be someone who does not doubt himself/herself. Can we define this concept in the basic hybrid language?

Example 1: Confidence

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 confident person to be someone who does not doubt 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:

$@_i \neg \langle \mathrm{DOUBT} \rangle i$ *(i does not doubt himself/herself)*

Example 1: Confidence

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 confident person to be someone who does not doubt 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:

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

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

Example 1: Confidence

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 confident person to be someone who does not doubt 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:

$@_i \neg \langle \text{DOUBT} \rangle i$ *(i does not doubt himself/herself)*

$@_j \neg \langle \text{DOUBT} \rangle j$ *(j does not doubt himself/herself)*

$@_k \neg \langle \text{DOUBT} \rangle k$ *(k does not doubt himself/herself)*

Example 1: Confidence

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 confident person to be someone who does not doubt 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:

$@_i \neg \langle \text{DOUBT} \rangle i$ *(i does not doubt himself/herself)*

$@_j \neg \langle \text{DOUBT} \rangle j$ *(j does not doubt himself/herself)*

$@_k \neg \langle \text{DOUBT} \rangle k$ *(k does not doubt himself/herself)*

None of these pins down the concept of confidence — only of "i being confident", "j being confident", "k being confident", and so on. We want to abstract away from the effects of particular nominals (constants).

Confidence via downarrow

With the aid of the downarrow binder, we can make this abstraction:

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

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 DOUBT relation.

To put it another way, it says: this person x (whoever x is) does not doubt himself/herself. In a sense, it's what linguists call a deictic definition (point-and-name).

The formula is true of precisely those states of our models (people) who do not doubt themselves; we have defined confidence

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 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 $\forall t(Rst \rightarrow Rts)$. 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 states that are locally reflected from ones that are not?

Well, we can try, but...

Note that $@_i \Box \Diamond i$ does not do what we want.

Well, we can try, but. . .

Note that $@_i \Box \Diamond i$ 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...

Note that $@_i\Box\Diamond i$ 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 a validity then we have an axiom that distinguishes symmetric models from all other models. But that's not what we want either.

Well, we can try, but...

Note that $@_i\Box\Diamond i$ 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 a validity then we have an axiom that distinguishes symmetric models from all other models. But that's not what we want either.
- We want a formula that classifies all the states in one model into those that are locally reflected and those that are not.

Locally reflected states via downarrow

Again, 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 is) is such that all its 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 in 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 in 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 in 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: $G(\mathrm{alarm} \rightarrow P\mathrm{problem})$ 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: $G(\text{reset} \rightarrow G(\text{alarm} \rightarrow P\text{problem}))$

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: $G(\text{reset} \rightarrow G(\text{alarm} \rightarrow P\text{problem}))$
- 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:

$$G(\text{reset} \rightarrow \downarrow x.G(\text{alarm} \rightarrow P(\text{problem} \wedge Px)))$$

Example 4: The *Until* operator

In this example we'll continue to think of the states of our models as points in 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

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

$$\mathit{Until}(\varphi, \psi) \;:=\; \downarrow x.F\downarrow y.(\varphi \wedge @_x G(Fy \rightarrow \psi)).$$

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 $= \{x, y, z \ldots\}$, 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:

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

- 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 $g : \mathrm{SVAR} \longrightarrow W$. (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 relativize 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

Some key clauses:

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

The fifth clause defines $\downarrow$ to be an binder: something binds variables to the state w at which the evaluation is being performed. The notation $g' \overset{x}{\sim} g$ 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.

$$\begin{aligned} \mathrm{ST}_x(z) &= (z = x) \\ \mathrm{ST}_x(@_s\varphi) &= \mathrm{ST}_s(\varphi) \\ \mathrm{ST}_x(\downarrow z.\varphi) &= \exists z(z = x \wedge \mathrm{ST}_x(\varphi)) \end{aligned}$$

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.

Standard translation: Alternative

But first recall that we gave an alternative translation for @ in Lecture 1. It's instructive to repeat this here, as it reveals what lies behind the "store" and "retrieve" interaction between $\downarrow$ and @:

$$\begin{aligned} \mathrm{ST}_x(z) &= (z = x) \\ \mathrm{ST}_x(@_s\varphi) &= \exists y(y = s \wedge \mathrm{ST}_y(\varphi)) \\ \mathrm{ST}_x(\downarrow z.\varphi) &= \exists z(z = x \wedge \mathrm{ST}_x(\varphi)) \end{aligned}$$

That is: $\downarrow$ "stores" the point of evaluation x as z, and continues translating with respect to x, whereas @ "retrieves" the value of s and continues the translation with respect to this retrieved value.

Tableau rules

We only need to make two changes. First, we need to let our previous tableau rules apply also when the subscript on @ is a state variable (that is: the previous rules now apply for both nominals and state variables).

Second, we add the following two rules to cope with $\downarrow$. In those rules, 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: $\downarrow x.x$

This (elegant!) sentence, "x names this spot!", is valid. And here is its tableaux proof:

Example: $\downarrow x.x$

This (elegant!) sentence, "x names this spot!", is valid. And here is its tableaux proof:

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

Example: $\downarrow x.x$

This (elegant!) sentence, "x names this spot!", is valid. And here is its tableaux proof:

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

Example: $\downarrow x.x$

This (elegant!) sentence, "x names this spot!", is valid. And here is its tableaux proof:

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	$@_i\neg\varphi[x \leftarrow i]$	$\downarrow$ 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	$@_i\neg\varphi[x \leftarrow i]$	$\downarrow$ rule on 3
5	$\neg @_i\varphi[x \leftarrow i]$	Propositional rule on 4
6	$@_i\varphi[x \leftarrow i]$	$\downarrow$ rule on 2
	$\bot_{5,6}$	

Tableau completeness

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

Nonetheless, 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 — and once again it's pure formulas that make it 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 $\downarrow$ (not just nominals, $\bot$ and $\top$), so we can define a lot more frame classes than before.
- Nonetheless, completeness is still automatic. Recall that if $@_i\varphi$ is a pure formula, whose nominals (if any) are $i, i_1, \ldots, i_n$, then we can turn it into the following tableau rule:

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

And there are interesting new axioms

For example, we cannot express the Church-Rosser (convergence) property using a pure formula of a hybrid language with one diamond. Incidentally, this is perhaps the simplest example of a first-order condition that you can't express with a pure formula, but you can express with ordinary propositional variables, namely with $\Diamond\Box p \to \Box\Diamond p$.

And there are interesting new axioms

For example, we cannot express the Church-Rosser (convergence) property using a pure formula of a hybrid language with one diamond. Incidentally, this is perhaps the simplest example of a first-order condition that you can't express with a pure formula, but you can express with ordinary propositional variables, namely with $\Diamond\Box p \rightarrow \Box\Diamond p$.

(Little aside: So what does $\Diamond\Box i \rightarrow \Box\Diamond i$ define...?)

And there are interesting new axioms

For example, we cannot express the Church-Rosser (convergence) property using a pure formula of a hybrid language with one diamond. Incidentally, this is perhaps the simplest example of a first-order condition that you can't express with a pure formula, but you can express with ordinary propositional variables, namely with $\Diamond\Box p \to \Box\Diamond p$.

(Little aside: So what does $\Diamond\Box i \to \Box\Diamond i$ define...?)

Though — as you have seen — you *can* express it if you have F and P:

$$F i \wedge F j \to F(i \wedge F P j).$$

And there are interesting new axioms

For example, we cannot express the Church-Rosser (convergence) property using a pure formula of a hybrid language with one diamond. Incidentally, this is perhaps the simplest example of a first-order condition that you can't express with a pure formula, but you can express with ordinary propositional variables, namely with $\Diamond\Box p \rightarrow \Box\Diamond p$.

(Little aside: So what does $\Diamond\Box i \rightarrow \Box\Diamond i$ define...?)

Though — as you have seen — you *can* express it if you have F and P:

$$F i \wedge F j \rightarrow F(i \wedge F P j).$$

With downarrow, it's easy:

$$\Diamond i \wedge \Diamond j \rightarrow \Diamond\Diamond{\downarrow}x.(@_i\Diamond x \wedge @_j\Diamond x)$$

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 described a couple of slides back). 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 $\downarrow$ has been added to the language.

What about axiomatics. . . ?

- It is extremely simply to axiomatize the downarrow binder.
- Yesterday we discussed the logic $\mathbf{K}_h$+RULES.
- If we add $\downarrow$ to the language, we get a (sound and complete) logic by adding one single axiom schema:

$$\text{DA:} \qquad \vdash @_i(\downarrow s.\varphi \leftrightarrow \varphi[s := i)$$

- Completeness easy to prove — because we know how build named models for $\mathbf{K}_h$+RULES.
- DA is very similar to the tableau rules for $\downarrow$.

But there is another way...

- But we can also start with $\mathbf{K}_h$. First, we can throw away NAME and PASTE and replace them by a standard generalization rule for downarrow: if $\vdash \varphi$ then $\vdash \downarrow s.\varphi$
- We get a (sound and complete) logic for $\downarrow$ by adding three axiom schemas:

  DA: $\vdash @_i(\downarrow s.\varphi \leftrightarrow \varphi[s := i)$

  $\text{Name}_\downarrow$: $\vdash \downarrow s.(s \rightarrow \varphi) \rightarrow \varphi$, where s not in φ.

  $\text{BG}_\downarrow$: $\vdash @_i \Box \downarrow s.@_i \Diamond s$

- $\text{BG}_\downarrow$ lets us build named models. Using it can derive PASTE, or a rule called BOUNDED GENERALISATION, which is related to the Natural Deduction rule we saw in Lecture 2.

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.

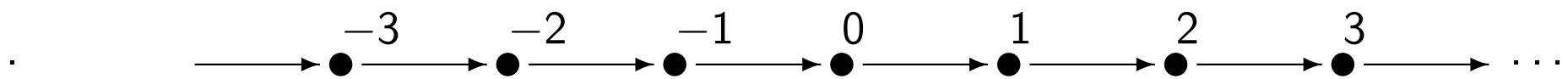
Towards the logic of locality

- 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 see, 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...

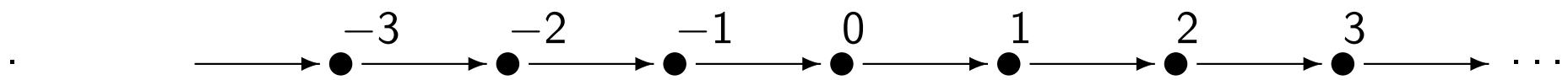
Submodels

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

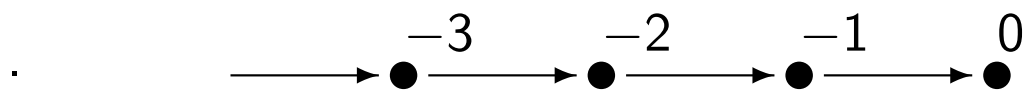


Submodels

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

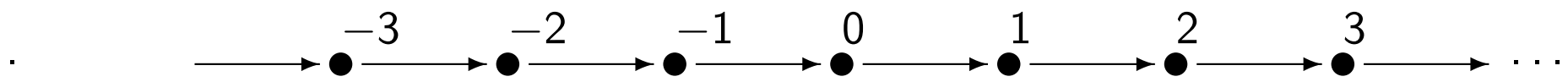


Suppose we form a submodel $\mathcal{M}^-$ of $\mathcal{M}$ by throwing away all the positive numbers, and restricting the original valuation (whatever it was) to the remaining numbers:

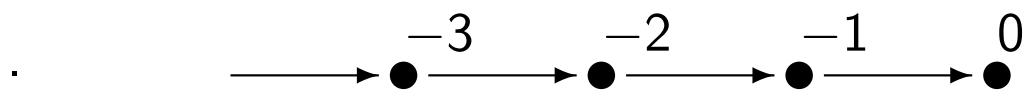


Submodels

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



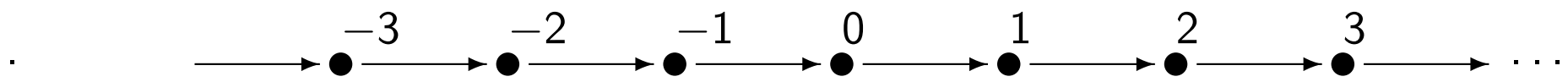
Suppose we form a submodel $\mathcal{M}^-$ of $\mathcal{M}$ by throwing away all the positive numbers, and restricting the original valuation (whatever it was) to the remaining numbers:



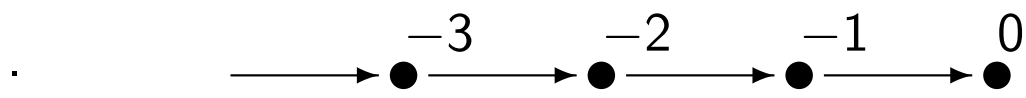
Can an orthodox modal language detect the difference between the two models?

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:

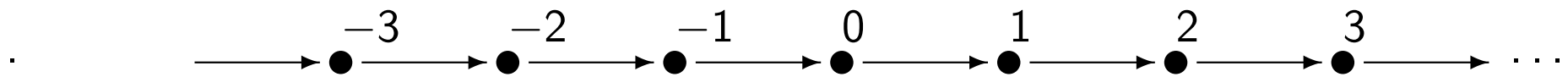


Can an orthodox modal language detect the difference between the two models?

Yes! $\mathcal{M}, 0 \models \Diamond\top$, but $\mathcal{M}^-, 0 \not\models \Diamond\top$

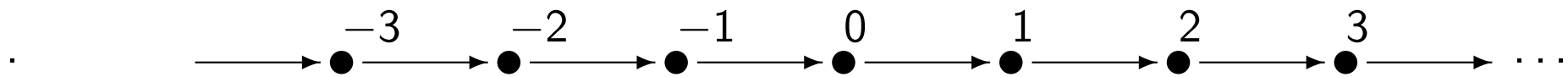
Another submodel

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

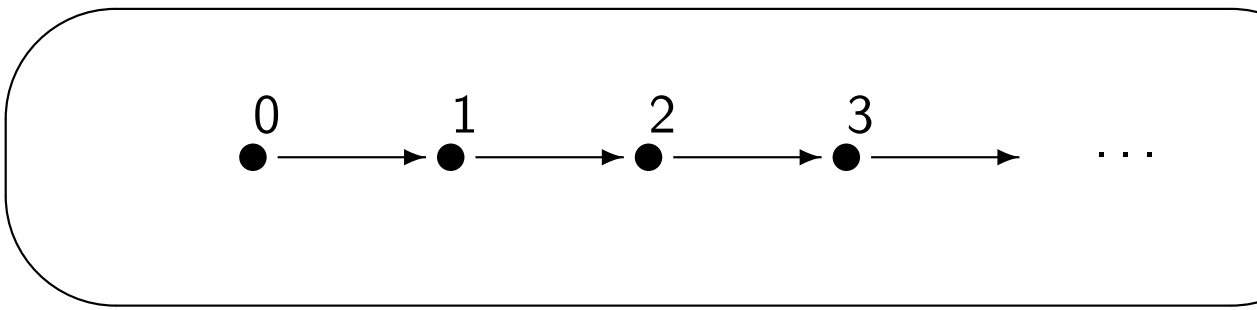


Another submodel

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

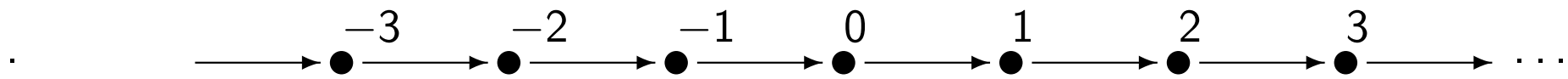


This time, suppose we form a submodel $\mathcal{M}^+$ of $\mathcal{M}$ by throwing away the negative numbers, and restricting the original valuation to what remains:

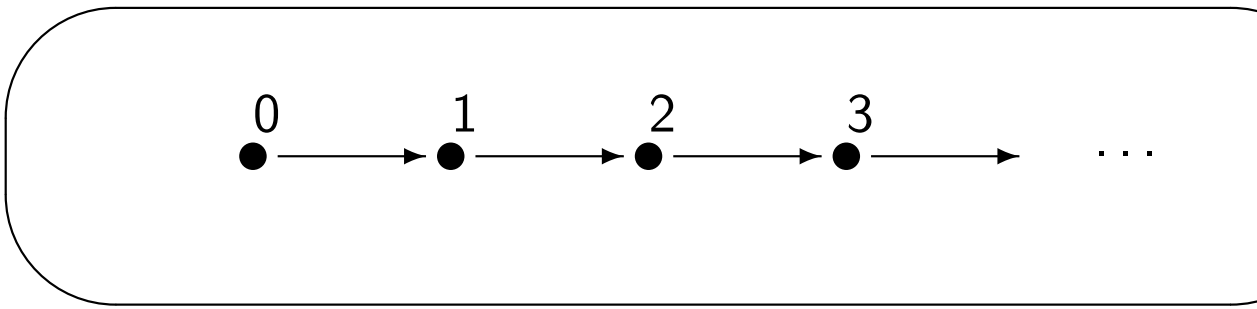


Another submodel

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



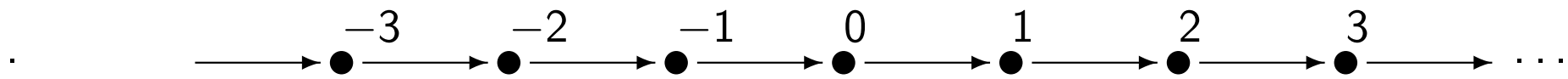
This time, suppose we form a submodel $\mathcal{M}^+$ of $\mathcal{M}$ by throwing away the negative numbers, and restricting the original valuation to what remains:



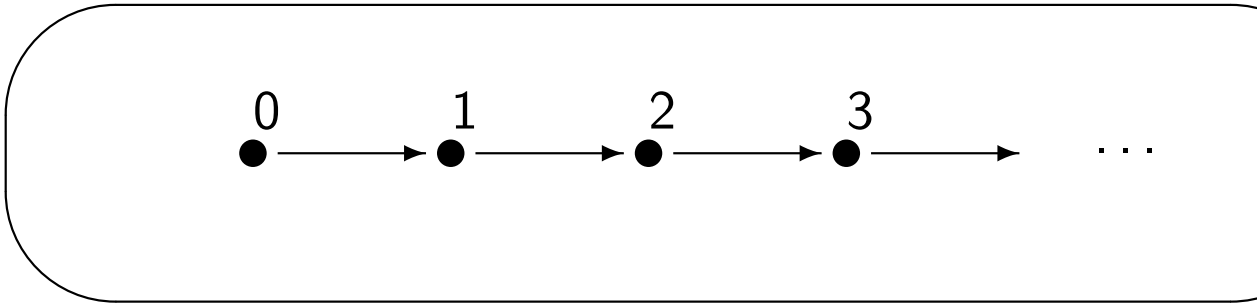
Can an orthodox modal language detect the difference between the two models?

Another submodel

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



This time, suppose we form a submodel $\mathcal{M}^+$ of $\mathcal{M}$ 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 models?

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

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 its 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 its 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 the submodel of $\mathcal{M}$ generated by w is the model $\mathcal{M}_w = (W_w, R_w, V_w)$ 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 the submodel of $\mathcal{M}$ generated by w is the model $\mathcal{M}_w = (W_w, R_w, V_w)$ 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:

$$\mathcal{M}, u \models \varphi \quad \text{iff} \quad \mathcal{M}_w, u \models \varphi.$$

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

Proof: By 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?

No!

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: Let $\mathcal{M}$ be a model, let $\mathcal{M}_w$ be any of its point generated submodels, and let 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 $\mathcal{M}_w$

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

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 point 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 point 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*.

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.

Interesting property, used in many places from Computer Science to the Philosophy of Science.

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 systematically repair interpolation failures.

Interpolation failure in basic hybrid logic

In Homework 2 you were asked to give 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).

$\models (\Diamond \mathrm{p} \wedge \Diamond \neg \mathrm{p}) \rightarrow \theta$ 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).
- 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 bisimulation argument shows that no interpolant exists.

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 hold 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 in a 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.

What is nice about the proof in this paper is it that it uses the model construction we saw yesterday. To prove interpolation in this way you use two Lindenbaum expansions simultaneously, and use the overlap in the common language to find the interpolant. It's a pretty proof method.

Hybrid logic with $\downarrow$ has interpolation

Theorem: Suppose we are working in a 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 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.

This paper uses a method that Melvin Fitting has popularized to actually calculate interpolants. Basically carry out tableau proof (with some extras) and then "read the interpolant off the tableau".

Moreover, it's minimal...

Actually, something even more interesting is the case. In

"Interpolation for extended modal languages", Balder ten Cate, *The Journal of Symbolic Logic*, 70(1), 223-234, 2005.

it is shown that extending the basic hybrid language by adding $\downarrow$ (as we have done today) is actually the *minimal* way of extending the basic hybrid language to obtain interpolation. That is: if we want interpolation in a hybrid logic, adding $\downarrow$ is the least we can do to obtain it. Using this result, Balder ten Cate goes on to prove a number of results about other extended modal logics.

The finite model property

- A language has the finite model property if any satisfiable formula in the language can be satisfied in 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 an unbounded strict total order. It is not hard to see that any unbounded strict total order is infinite. So first-order logic lacks 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 spypoint s (the spypoint) that can see strict unbounded total order (see "Hybrid Languages" Blackburn and Seligman, *Journal of Logic, Language and Information* 4, 251-272, 1995).

A spypoint argument

Consider what any model of the following formula must contain:

$$@_s\Box\Box \downarrow x.@_s\Diamond x$$

$$\wedge \quad @_s\Diamond\neg s$$

$$\wedge \quad @_s\Box\Diamond\top$$

$$\wedge \quad @_s\Box \downarrow x.\neg\Diamond x$$

$$\wedge \quad @_s\Box \downarrow x.\Box\Box \downarrow y.@_x\Diamond y$$

This formula has some obvious models.

Moreover, any model for this formula must contain a spypoint 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 $\downarrow$ is undecidable

- We have stepped over an important boundary: adding $\downarrow$ 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: we use the spypoint as a vantage point surveying a coding of an undecidable problem. (See "Hybrid Languages" Blackburn and Seligman, *Journal of Logic, Language and Information* 4, 251-272, 1995 for the original proof and "Hybrid Logics: Characterization, Interpolation and Complexity", Areces, Blackburn and Marx, *Journal of Symbolic Logic*, 66:977-1009, 2001 for a sharper result.)

Mapping the border...

When (and how) does undecidability kick in? Which syntactic and semantic restrictions restore decidability? For some answers see:

"Narcissists, Stepmothers and Spies", Maarten Marx, *Proceedings of the International Workshop on Description Logics*, vol 53, CEUR, 2002.

"On the complexity of hybrid logics with binders", Balder ten Cate Massimo Franceschet, *Proceedings of the 19th International Workshop on Computer Science Logic*, Lecture Notes in Computer Science vol. 3634, pp. 339-354, Springer.

"Complexity of hybrid logics over transitive frames", Martin Mundhenk, Thomas Schneider, Thomas Schwentick, Volker Weber, *Journal of Applied Logic*, 8(4), pp. 422-440, 2010.

"The complexity of satisfiability for fragments of hybrid logic — Part I", Arne Meier, Martin Mundhenk, Thomas Schneider, Michael Thomas, Volker Weber, Journal of Applied Logic 8(4), pp. 409-421, 2010.

Summing up . . .

- We motivated the idea of binding variables to states locally, and introduced $\downarrow$ 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.
- But although local, the existence of infinite models can be forced, and the system is undecidable.