Modal Logics: A Modern Perspective

Carlos Areces
carlos.areces@gmail.com

Spring Term 2018
Stanford

Generalities

- This is

  “Modal Logic: A Modern Perspective”

  an advanced course on Modal Logics.

Generalities

- This is

  "Modal Logic: A Modern Perspective"

  an advanced course on Modal Logics.
  - Some particular Modal Logics: Hybrid Logics, Dynamic Epistemic Logics, Description Logics, etc.

Generalities

- This is

  "Modal Logic: A Modern Perspective"

  an advanced course on Modal Logics.
  - Some particular Modal Logics: Hybrid Logics, Dynamic Epistemic Logics, Description Logics, etc.
  - Focus on expressivity, complexity, axiomatics, …

Generalities

- This is

  "Modal Logic: A Modern Perspective"

  an advanced course on Modal Logics.
  - Some particular Modal Logics: Hybrid Logics, Dynamic Epistemic Logics, Description Logics, etc.
  - Focus on expressivity, complexity, axiomatics, …
- Previous knowledge assumed

Generalities

- This is

  "Modal Logic: A Modern Perspective"

  an advanced course on Modal Logics.
  - Some particular Modal Logics: Hybrid Logics, Dynamic Epistemic Logics, Description Logics, etc.
  - Focus on expressivity, complexity, axiomatics, …
- Previous knowledge assumed
  - Good knowledge of Propositional Logic and First-Order Logic

Generalities

- This is

  "Modal Logic: A Modern Perspective"

  an advanced course on Modal Logics.
  - Some particular Modal Logics: Hybrid Logics, Dynamic Epistemic Logics, Description Logics, etc.
  - Focus on expressivity, complexity, axiomatics, ...
- Previous knowledge assumed
  - Good knowledge of Propositional Logic and First-Order Logic
  - (Some) Previous knowledge of Modal Logic

What we can cover

- **Unit I:** Introduction (Aims and Evaluation, Methodology). Recap of Propositional and First Order Logic. Basic Modal Logic (Syntax and Semantics). Motivation, Examples, Applications. Other Modal Operators.
- **Unit II:** Modal Logics as Fragments of Classical Logics. The Standard Translation. Transference of Results (Decidability via Translation). Optimized Translations. Automated Reasoning via Translations.
- **Unit III:** Hybrid Logics. Decidable Hybrid Logics (Tableau Calculi). The $\downarrow$ and @ operators and Undecidability. Axiomatizations (Henkin Models). Hybrid XPath (XPath as a query language, Axiomatization, Tableau Calculus).
- **Unit IV:** Description Logics. Web Ontologies and the Semantic Web. Knowledge Bases. A-Box and T-Box Reasoning (Tableau Calculi, Termination).
- **Unit V:** Dynamic Modal Logics. Epistemic Logics (Applications and Limitations). Dynamic Epistemic Logics (Public Announcement Logic, Action Modal Logics, Expressive Power, Decidability). Other Dynamic Modal Operators (Fragments of First Order Logic, Undecidability).

The Course

- Timetable:

The Course

- Timetable:
  - Tuesday & Thursday: 10.30 to 11.50.

The Course

- Timetable:
  - Tuesday & Thursday: 10.30 to 11.50.
- Evaluation: Not sure yet, (I'm not from here)

The Course

- Timetable:
  - Tuesday & Thursday: 10.30 to 11.50.
- Evaluation: Not sure yet, (I'm not from here)
- Website of the course

  `http://cs.famaf.unc.edu.ar/~careces/ml18`

  Homework, slides from the course, additional reading material will be posted here.

The Course

- Timetable:
  - Tuesday & Thursday: 10.30 to 11.50.
- Evaluation: Not sure yet, (I'm not from here)
- Website of the course

  `http://cs.famaf.unc.edu.ar/~careces/ml18`

  Homework, slides from the course, additional reading material will be posted here.
- Textbook: there is no textbook for the course.

The Course

- Timetable:
  - Tuesday & Thursday: 10.30 to 11.50.
- Evaluation: Not sure yet, (I'm not from here)
- Website of the course

  `http://cs.famaf.unc.edu.ar/~careces/ml18`

  Homework, slides from the course, additional reading material will be posted here.
- Textbook: there is no textbook for the course.
  - Modal Logic. Blackburn, de Rijke, & Venema

The Course

- Timetable:
  - Tuesday & Thursday: 10.30 to 11.50.
- Evaluation: Not sure yet, (I'm not from here)
- Website of the course

  `http://cs.famaf.unc.edu.ar/~careces/ml18`

  Homework, slides from the course, additional reading material will be posted here.
- Textbook: there is no textbook for the course.
  - Modal Logic. Blackburn, de Rijke, & Venema
  - First Steps in Modal Logic. Popkorn

Myself

- I am Carlos Areces (hi!), from Argentina.
- Email: `carlos.areces@gmail.com`
- Web: `https://cs.famaf.unc.edu.ar/~careces/`
- Studied CS at Universidad de Buenos Aires, then I did a PhD degree in Logic at ILLC Amsterdam (supervisor Maarten de Rijke, promotor Johan van Benthem), then a postdoc there, then 7 years at INRIA-Nancy in France.

Myself

- I am Carlos Areces (hi!), from Argentina.
- Email: `carlos.areces@gmail.com`
- Web: `https://cs.famaf.unc.edu.ar/~careces/`
- Studied CS at Universidad de Buenos Aires, then I did a PhD degree in Logic at ILLC Amsterdam (supervisor Maarten de Rijke, promotor Johan van Benthem), then a postdoc there, then 7 years at INRIA-Nancy in France.
- Currently a professor at Universidad Nacional de Córdoba, Argentina.

Myself

- I am Carlos Areces (hi!), from Argentina.
- Email: `carlos.areces@gmail.com`
- Web: `https://cs.famaf.unc.edu.ar/~careces/`
- Studied CS at Universidad de Buenos Aires, then I did a PhD degree in Logic at ILLC Amsterdam (supervisor Maarten de Rijke, promotor Johan van Benthem), then a postdoc there, then 7 years at INRIA-Nancy in France.
- Currently a professor at Universidad Nacional de Córdoba, Argentina.
- Now just arrived at Stanford University (and it is great!).

A quick test

A quick test

What can you tell me about the following?

- $\Box(p \to q) \to (\Box p \to \Box q)$

A quick test

What can you tell me about the following?

- $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$
- Maximal Consisten Set (MCS)

A quick test

What can you tell me about the following?

- $\Box(p \to q) \to (\Box p \to \Box q)$
- Maximal Consisten Set (MCS)
- Bisimulation

A quick test

What can you tell me about the following?

- $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$
- Maximal Consisten Set (MCS)
- Bisimulation
- EF-Games

A quick test

What can you tell me about the following?

- $\Box(p \to q) \to (\Box p \to \Box q)$
- Maximal Consisten Set (MCS)
- Bisimulation
- EF-Games
- Tableaux

A quick test

What can you tell me about the following?

- $\Box(p \to q) \to (\Box p \to \Box q)$
- Maximal Consisten Set (MCS)
- Bisimulation
- EF-Games
- Tableaux
- PSPace-complete

A quick test

What can you tell me about the following?

- $\Box(p \to q) \to (\Box p \to \Box q)$
- Maximal Consisten Set (MCS)
- Bisimulation
- EF-Games
- Tableaux
- PSPace-complete
- Tiling

A quick test

What can you tell me about the following?

- $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$
- Maximal Consisten Set (MCS)
- Bisimulation
- EF-Games
- Tableaux
- PSPace-complete
- Tiling
- Muddy children

A quick test

What can you tell me about the following?

- $\Box(p \to q) \to (\Box p \to \Box q)$
- Maximal Consisten Set (MCS)
- Bisimulation
- EF-Games
- Tableaux
- PSPace-complete
- Tiling
- Muddy children
- XML

A quick test

What can you tell me about the following?

- $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$
- Maximal Consisten Set (MCS)
- Bisimulation
- EF-Games
- Tableaux
- PSPace-complete
- Tiling
- Muddy children
- XML
- Tree automata

Why “Logics”

Why "Logics"

- Usually, Logic is first-order logic

$$\forall x.(\text{bird}(x) \wedge \text{early}(x) \rightarrow \exists y.(\text{worm}(y) \wedge \text{catches}(x, y)))$$

Why “Logics”

- Usually, Logic is first-order logic

$$\forall x.(\text{bird}(x) \wedge \text{early}(x) \rightarrow \exists y.(\text{worm}(y) \wedge \text{catches}(x, y)))$$

- Of course, there is also propositional logic:

$$\text{toBe} \vee \neg\text{toBe}$$

Why "Logics"

- Usually, Logic is first-order logic

$$\forall x.(\mathrm{bird}(x) \wedge \mathrm{early}(x) \rightarrow \exists y.(\mathrm{worm}(y) \wedge \mathrm{catches}(x, y)))$$

- Of course, there is also propositional logic:

$$\mathrm{toBe} \vee \neg \mathrm{toBe}$$

- How many logics are there?

Why “Logics”

- Usually, Logic is first-order logic

$$\forall x.(\mathrm{bird}(x) \wedge \mathrm{early}(x) \rightarrow \exists y.(\mathrm{worm}(y) \wedge \mathrm{catches}(x, y)))$$

- Of course, there is also propositional logic:

$$\mathrm{toBe} \vee \neg \mathrm{toBe}$$

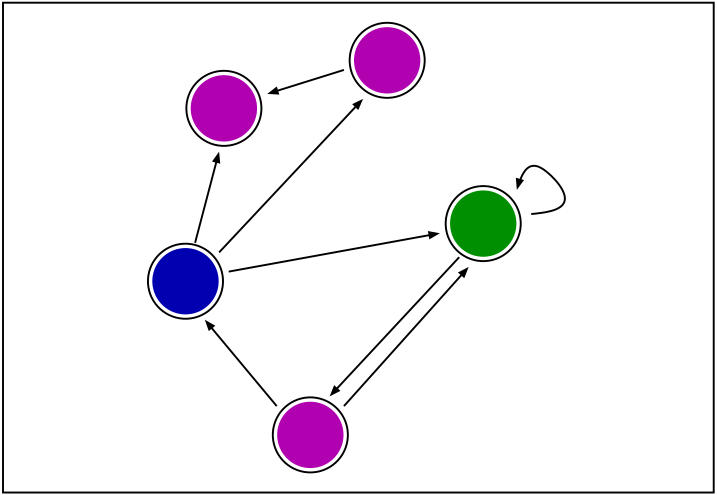
- How many logics are there?
- In this course we are going to
  - Discuss different modal logics, from different perspectives
  - Take, mainly, a model theoretic approach (e.g., bisimulations)
  - Discuss computational issues (e.g., complexity)

Simple structures for simple languages

Think of a colored graph:

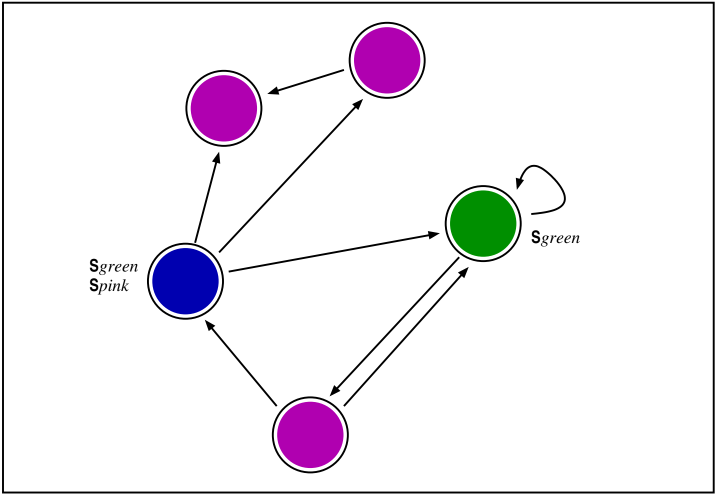
Simple structures for simple languages

Think of a colored graph:



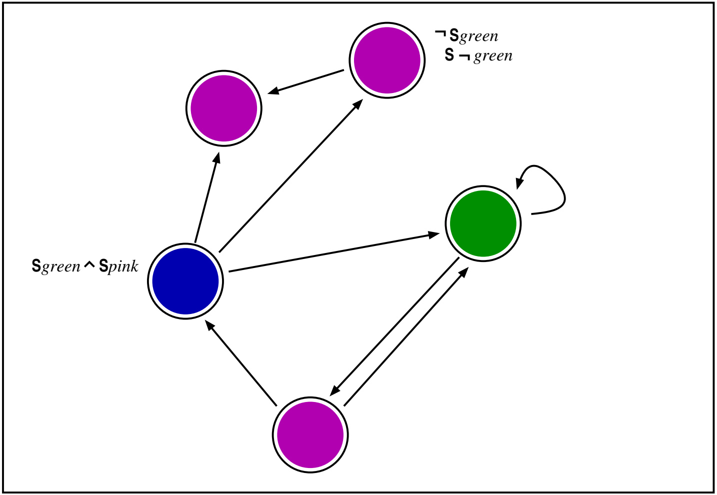
Simple structures for simple languages

Think of a colored graph:



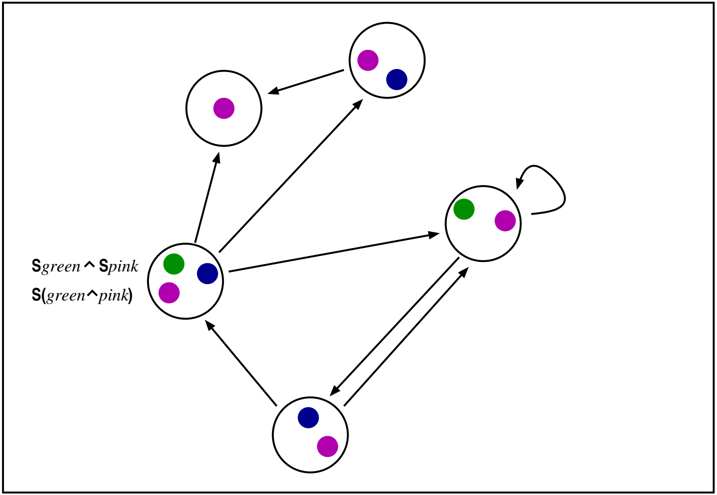
Simple structures for simple languages

Think of a colored graph:



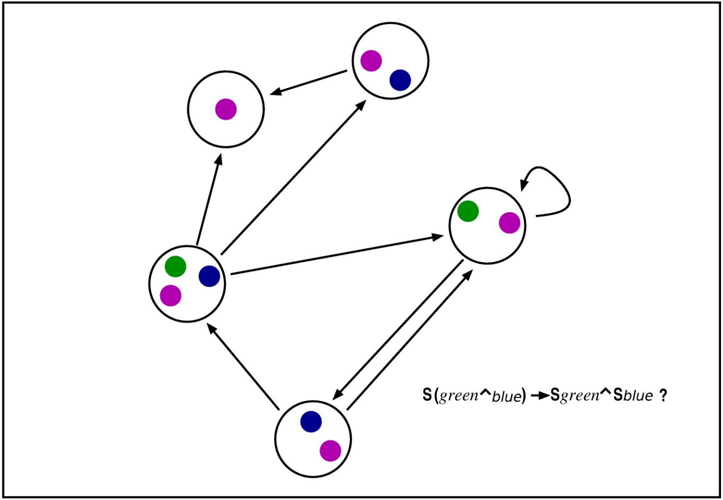
Simple structures for simple languages

Think of a colored graph:



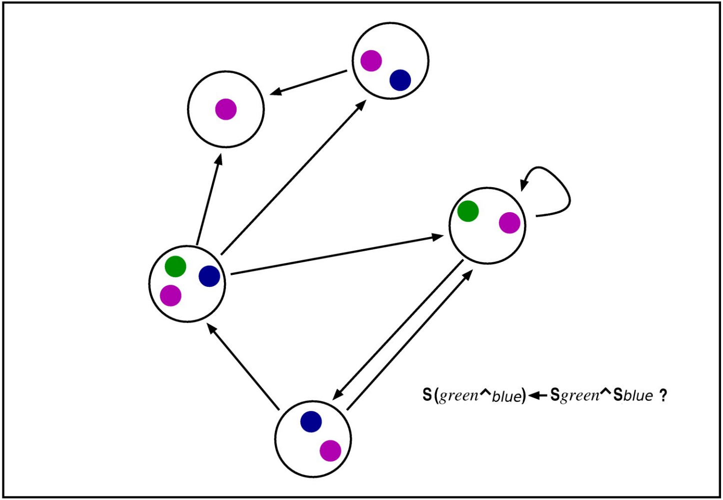
Simple structures for simple languages

Think of a colored graph:



Simple structures for simple languages

Think of a colored graph:



Simple structures for simple languages

- That was a *modal logic*.

Simple structures for simple languages

- That was a *modal logic*.
- It looks like a good language to talk about colored graphs. Useful?

Simple structures for simple languages

- That was a *modal logic*.
- It looks like a good language to talk about colored graphs. Useful?
- First advantage:
  - To decide weather a first-order formula is satisfiable, is undecidable.
  - In the modal logic we just presented, it is computable! (actually, PSPACE-complete)

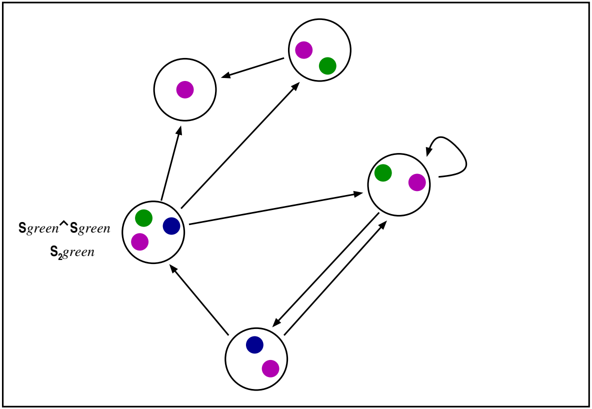
A family of languages

A family of languages

- Sometimes, the language we just introduced is not the correct one:

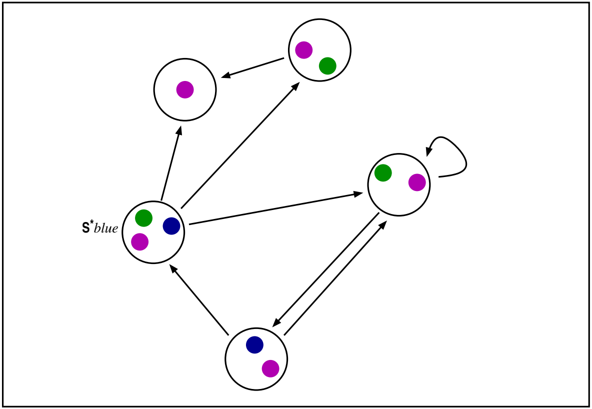
A family of languages

- Sometimes, the language we just introduced is not the correct one:



A family of languages

- Sometimes, the language we just introduced is not the correct one:



A family of languages

- Sometimes, the language we just introduced is not the correct one:
- Modal Logic studies a wide spectrum of possible languages that can be used to describe relational structures.

A family of languages

- Sometimes, the language we just introduced is not the correct one:
- Modal Logic studies a wide spectrum of possible languages that can be used to describe relational structures.
- Some questions we can ask:
  - Which are the expressivity limits of these languages?
  - Can we define inference algorithm for these languages?
  - How efficient are they?

A family of languages

- Sometimes, the language we just introduced is not the correct one:
- Modal Logic studies a wide spectrum of possible languages that can be used to describe relational structures.
- Some questions we can ask:
  - Which are the expressivity limits of these languages?
  - Can we define inference algorithm for these languages?
  - How efficient are they?
- An alternative perspective is to look at them from the perspective of logic engineering. Give a problem that requires inference:
  - Which is the best language we can use? (e,g. the easiest to use)
  - Which logic has good inference algorithms?
  - Which operators do I need?

Possible applications

Possible applications

- Modal Logics are used is a variety of areas

Possible applications

- Modal Logics are used is a variety of areas
  - Software and Hardware Verification.
  - Knowledge representation.
  - Computational linguistics.
  - Cryptography.
  - Artificial Intelligence.
  - Philosophy.
  - …

Possible applications

- Modal Logics are used is a variety of areas
  - Software and Hardware Verification.
  - Knowledge representation.
  - Computational linguistics.
  - Cryptography.
  - Artificial Intelligence.
  - Philosophy.
  - …
- **Why?**

Possible applications

- Modal Logics are used is a variety of areas
  - Software and Hardware Verification.
  - Knowledge representation.
  - Computational linguistics.
  - Cryptography.
  - Artificial Intelligence.
  - Philosophy.
  - ...
- **Why?** Because many things can be represented as colored graphs (i.e., relational structures).

So much better than classical logic

So much better than classical logic

- Actually, we are doing classical logic.

So much better than classical logic

- Actually, we are doing classical logic.
- The languages we are going to study can be seen as fragments of first (or second) order logic. We are just carefully choosing which fragment we need for a particular application

So much better than classical logic

- Actually, we are doing classical logic.
- The languages we are going to study can be seen as fragments of first (or second) order logic. We are just carefully choosing which fragment we need for a particular application
- This is, actually, how I think of modal languages. As tools to investigate interesting fragments of better known classical logics.

So much better than classical logic

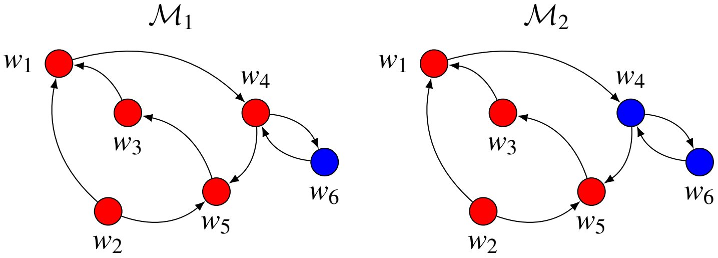
- Actually, we are doing classical logic.
- The languages we are going to study can be seen as fragments of first (or second) order logic. We are just carefully choosing which fragment we need for a particular application
- This is, actually, how I think of modal languages. As tools to investigate interesting fragments of better known classical logics.
- Where "interesting means
  - Decidable, expressive, of "low" complexity, modular, etc.

First-order logic

- The notion of truth in first-order logic is related to sentences, i.e., formulas without free variables.

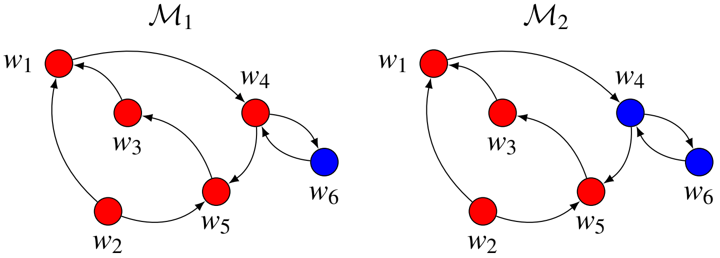
First-order logic

- The notion of truth in first-order logic is related to sentences, i.e., formulas without free variables.
- Let φ be an arbitrary sentence, and $\mathcal{M}$ a first-order model (in the signature of φ). Then φ will either be true or false in (all) $\mathcal{M}$. We cannot talk of a particular part of the model with φ.



First-order logic

- The notion of truth in first-order logic is related to sentences, i.e., formulas without free variables.
- Let φ be an arbitrary sentence, and $\mathcal{M}$ a first-order model (in the signature of φ). Then φ will either be true or false in (all) $\mathcal{M}$. We cannot talk of a particular part of the model with φ.

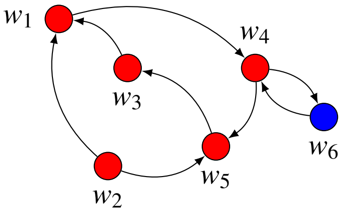


- $\mathcal{M}_1 \models \forall x.(red(x) \rightarrow (\exists y.xRy \wedge red(y)))$

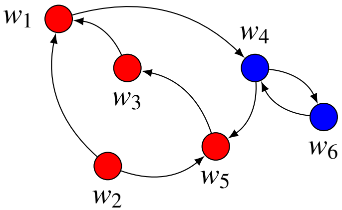
First-order logic

- The notion of truth in first-order logic is related to sentences, i.e., formulas without free variables.
- Let φ be an arbitrary sentence, and $\mathcal{M}$ a first-order model (in the signature of φ). Then φ will either be true or false in (all) $\mathcal{M}$. We cannot talk of a particular part of the model with φ.

$\mathcal{M}_1$



$\mathcal{M}_2$



- $\mathcal{M}_1 \models \forall x.(red(x) \to (\exists y.xRy \land red(y)))$
- $\mathcal{M}_2 \not\models \forall x.(red(x) \to (\exists y.xRy \land red(y)))$

First-order logic

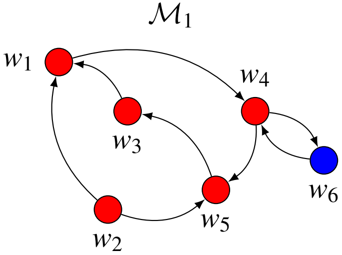
- This means that the extension (meaning) of the a sentence in FOL is either the empty set or the whole model.

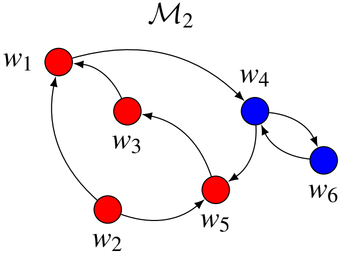
First-order logic

- This means that the extension (meaning) of the a sentence in FOL is either the empty set or the whole model.
- How do we talk of parts of the model in FOL?

First-order logic

- This means that the extension (meaning) of the a sentence in FOL is either the empty set or the whole model.
- How do we talk of parts of the model in FOL?
- We use formulas with free variables:

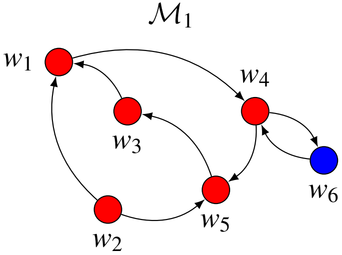




- $sRed(x) = red(x) \rightarrow (\exists y.xRy \wedge red(y))$

First-order logic

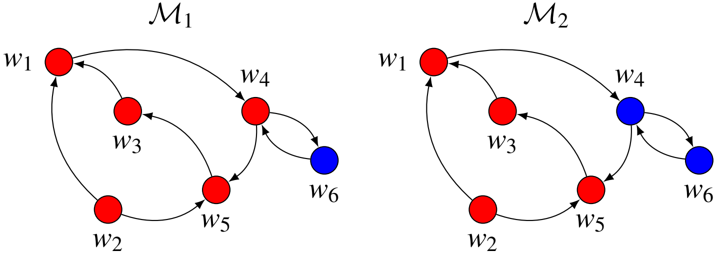
- This means that the extension (meaning) of the a sentence in FOL is either the empty set or the whole model.
- How do we talk of parts of the model in FOL?
- We use formulas with free variables:



- $sRed(x) = red(x) \to (\exists y.xRy \wedge red(y))$
- $[sRed(x)]^{\mathcal{M}_1} = [red(x) \to (\exists y.xRy \wedge red(y))]^{\mathcal{M}_1} =$

First-order logic

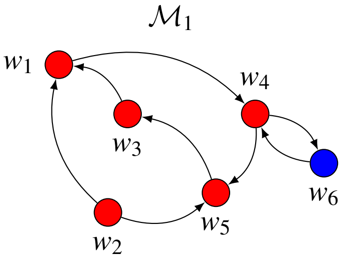
- This means that the extension (meaning) of the a sentence in FOL is either the empty set or the whole model.
- How do we talk of parts of the model in FOL?
- We use formulas with free variables:



- $sRed(x) = red(x) \rightarrow (\exists y.xRy \wedge red(y))$
- $[sRed(x)]^{\mathcal{M}_1} = [red(x) \rightarrow (\exists y.xRy \wedge red(y))]^{\mathcal{M}_1} = \{w_1, \ldots, w_6\}$

First-order logic

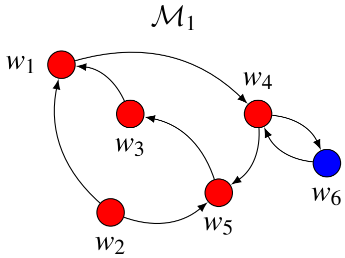
- This means that the extension (meaning) of the a sentence in FOL is either the empty set or the whole model.
- How do we talk of parts of the model in FOL?
- We use formulas with free variables:



- $sRed(x) = red(x) \rightarrow (\exists y.xRy \wedge red(y))$
- $[sRed(x)]^{\mathcal{M}_1} = [red(x) \rightarrow (\exists y.xRy \wedge red(y))]^{\mathcal{M}_1} = \{w_1, \ldots, w_6\}$
- $[sRed(x)]^{\mathcal{M}_2} = [red(x) \rightarrow (\exists y.xRy \wedge red(y))]^{\mathcal{M}_2} =$

First-order logic

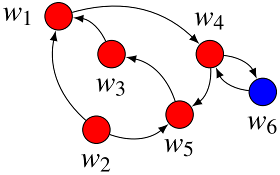
- This means that the extension (meaning) of the a sentence in FOL is either the empty set or the whole model.
- How do we talk of parts of the model in FOL?
- We use formulas with free variables:



- $sRed(x) = red(x) \to (\exists y.xRy \wedge red(y))$
- $[sRed(x)]^{\mathcal{M}_1} = [red(x) \to (\exists y.xRy \wedge red(y))]^{\mathcal{M}_1} = \{w_1, \ldots, w_6\}$
- $[sRed(x)]^{\mathcal{M}_2} = [red(x) \to (\exists y.xRy \wedge red(y))]^{\mathcal{M}_2} = \{w_2, \ldots, w_6\}$

Internal Perspective

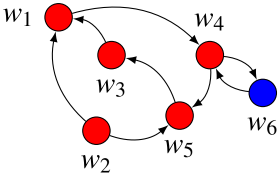
- Things are easier using a modal language:



$$\mathcal{M}_1, w_1 \models \mathit{red} \rightarrow \langle \mathit{see} \rangle \mathit{red}$$

Internal Perspective

- Things are easier using a modal language:

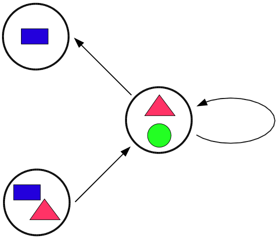


$$\mathcal{M}_1, w_1 \models red \rightarrow \langle see \rangle red$$

- $[red \rightarrow \langle see \rangle red]^{\mathcal{M}_1} = \{w_1, \ldots, w_6\}$
- $[red \rightarrow \langle see \rangle red]^{\mathcal{M}_2} = \{w_2, \ldots, w_6\}$

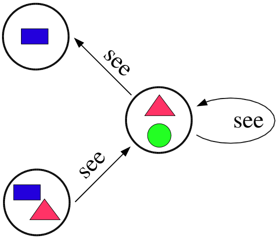
Describing Labeled Graphs

Consider now a graph with figures inside their nodes:



Describing Labeled Graphs

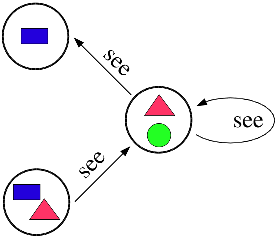
Consider now a graph with figures inside their nodes:



We want to describe what can be seen from a node.

Describing Labeled Graphs

Consider now a graph with figures inside their nodes:



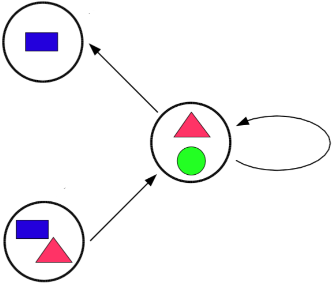
We want to describe what can be seen from a node.

From the perspective of a node n, the meaning of the classical modal operators would be:

- $\langle see \rangle x$ = "n can see figure x in one neighbor.
- $[see]x$ = "n can see figure x in all neighbors.

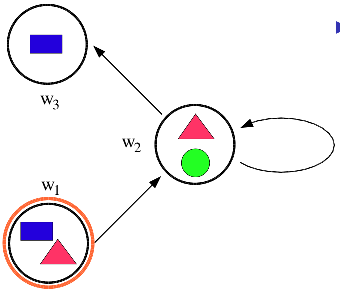
Describing Labeled Graphs

Now we can "query" the model:



Describing Labeled Graphs

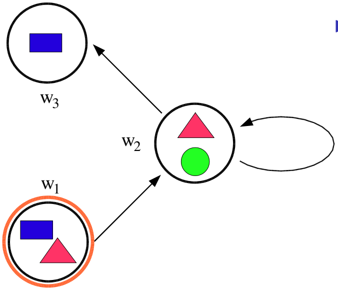
Now we can "query" the model:



- Can w_1 see a rectangle?

Describing Labeled Graphs

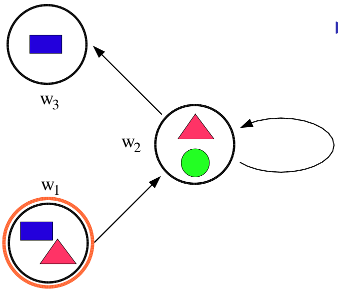
Now we can "query" the model:



- Can w_1 see a rectangle?
  $\mathcal{M}, w_1 \models \langle see \rangle rectangle$?

Describing Labeled Graphs

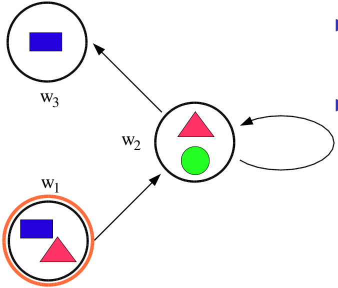
Now we can "query" the model:



- Can w_1 see a rectangle?
  $\mathcal{M}, w_1 \models \langle see \rangle rectangle$? NO

Describing Labeled Graphs

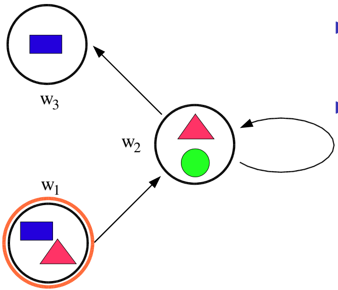
Now we can "query" the model:



- Can w_1 see a rectangle?
  $\mathcal{M}, w_1 \models \langle see \rangle rectangle$? NO
- …and in two "steps?

Describing Labeled Graphs

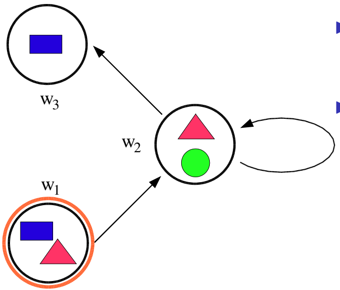
Now we can "query" the model:



- Can w_1 see a rectangle?
  $\mathcal{M}, w_1 \models \langle see \rangle rectangle$? NO
- …and in two "steps?
  $\mathcal{M}, w_1 \models \langle see \rangle \langle see \rangle rectangle$?

Describing Labeled Graphs

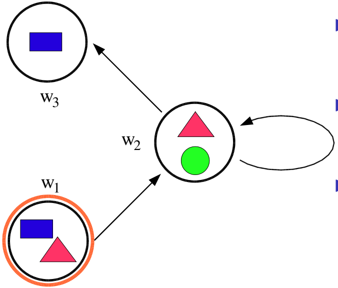
Now we can "query" the model:



- Can w_1 see a rectangle?
  $\mathcal{M}, w_1 \models \langle see \rangle rectangle$? NO
- ...and in two "steps?
  $\mathcal{M}, w_1 \models \langle see \rangle \langle see \rangle rectangle$? YES

Describing Labeled Graphs

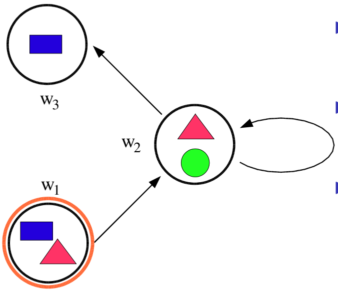
Now we can "query" the model:



- Can w_1 see a rectangle?
  $\mathcal{M}, w_1 \models \langle see \rangle rectangle$? NO
- ...and in two "steps?
  $\mathcal{M}, w_1 \models \langle see \rangle \langle see \rangle rectangle$? YES
- Can w_1 see a circle and a triangle

Describing Labeled Graphs

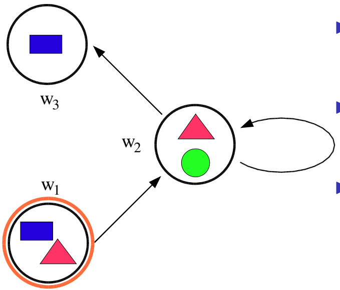
Now we can “query” the model:



- Can w_1 see a rectangle?
  $\mathcal{M}, w_1 \models \langle see \rangle rectangle$? NO
- …and in two “steps?
  $\mathcal{M}, w_1 \models \langle see \rangle \langle see \rangle rectangle$? YES
- Can w_1 see a circle and a triangle
  $\mathcal{M}, w_1 \models \langle see \rangle circle \wedge \langle see \rangle triangle$?

Describing Labeled Graphs

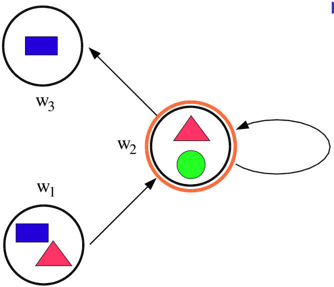
Now we can "query" the model:



- Can w_1 see a rectangle?
  $\mathcal{M}, w_1 \models \langle see \rangle rectangle$? NO
- …and in two "steps?
  $\mathcal{M}, w_1 \models \langle see \rangle \langle see \rangle rectangle$? YES
- Can w_1 see a circle and a triangle
  $\mathcal{M}, w_1 \models \langle see \rangle circle \wedge \langle see \rangle triangle$? YES

Describing Labeled Graphs

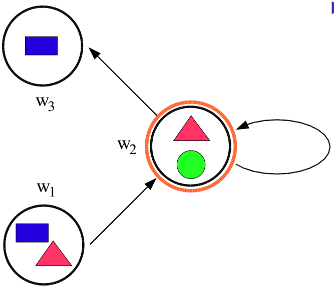
Now we can "query" the model:



- Can w_2 see a circle and a rectangle?
  $\mathcal{M}, w_2 \models \langle see \rangle circle \wedge \langle see \rangle rectangle$?

Describing Labeled Graphs

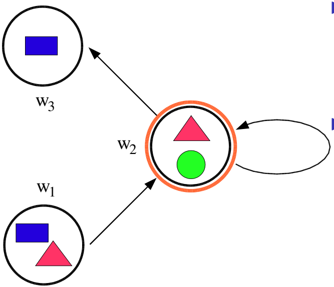
Now we can “query” the model:



- Can w_2 see a circle and a rectangle?
  $\mathcal{M}, w_2 \models \langle see \rangle circle \wedge \langle see \rangle rectangle$?
  YES

Describing Labeled Graphs

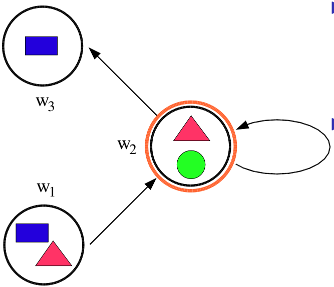
Now we can "query" the model:



- Can w_2 see a circle and a rectangle?
  $\mathcal{M}, w_2 \models \langle see \rangle circle \wedge \langle see \rangle rectangle$?
  YES
- …in the same neighbor?
  $\mathcal{M}, w_2 \models \langle see \rangle (circle \wedge rectangle)$?

Describing Labeled Graphs

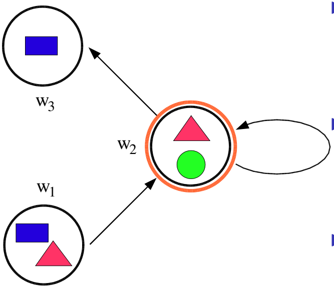
Now we can "query" the model:



- Can w_2 see a circle and a rectangle?
  $\mathcal{M}, w_2 \models \langle see \rangle circle \wedge \langle see \rangle rectangle$?
  YES
- …in the same neighbor?
  $\mathcal{M}, w_2 \models \langle see \rangle (circle \wedge rectangle)$?
  NO

Describing Labeled Graphs

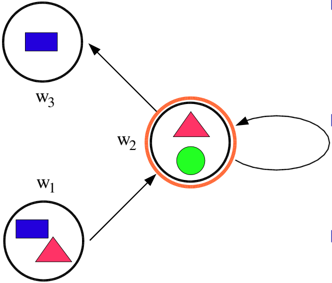
Now we can "query" the model:



- Can w_2 see a circle and a rectangle?
  $\mathcal{M}, w_2 \models \langle see \rangle circle \wedge \langle see \rangle rectangle$?
  YES
- …in the same neighbor?
  $\mathcal{M}, w_2 \models \langle see \rangle (circle \wedge rectangle)$?
  NO
- Can w_2 see circles in all neighbors?
  $\mathcal{M}, w_2 \models [see] circle$?

Describing Labeled Graphs

Now we can "query" the model:



- Can w_2 see a circle and a rectangle?
  $\mathcal{M}, w_2 \models \langle see \rangle circle \wedge \langle see \rangle rectangle$?
  YES
- …in the same neighbor?
  $\mathcal{M}, w_2 \models \langle see \rangle (circle \wedge rectangle)$?
  NO
- Can w_2 see circles in all neighbors?
  $\mathcal{M}, w_2 \models [see] circle$? NO

Describing Labeled Graphs

We can also thing of modal languages as a tool to describe processes.

Describing Labeled Graphs

We can also thing of modal languages as a tool to describe processes.

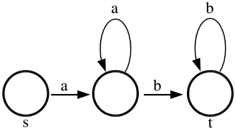
- E.g., we can see stats in the graphs as states (e.g., the states of a computation).

Describing Labeled Graphs

We can also thing of modal languages as a tool to describe processes.

- E.g., we can see stats in the graphs as states (e.g., the states of a computation).
- And consider the binary relations as actions that transform one state into another.

  Let's see an example:

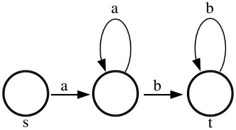


Describing Labeled Graphs

We can also thing of modal languages as a tool to describe processes.

- E.g., we can see stats in the graphs as states (e.g., the states of a computation).
- And consider the binary relations as actions that transform one state into another.

  Let's see an example:



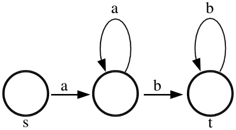
- The picture shows a finite automata for the language $a^n b^m$.

Describing Labeled Graphs

We can also thing of modal languages as a tool to describe processes.

- E.g., we can see stats in the graphs as states (e.g., the states of a computation).
- And consider the binary relations as actions that transform one state into another.

  Let's see an example:



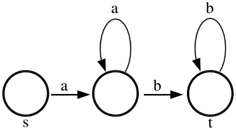
- The picture shows a finite automata for the language $a^n b^m$.
- In this case we have to modal operators $\langle a \rangle$ and $\langle b \rangle$

Describing Labeled Graphs

We can also thing of modal languages as a tool to describe processes.

- E.g., we can see stats in the graphs as states (e.g., the states of a computation).
- And consider the binary relations as actions that transform one state into another.

  Let's see an example:



- The picture shows a finite automata for the language $a^n b^m$.
- In this case we have to modal operators $\langle a \rangle$ and $\langle b \rangle$
- All formulas of the form

$$\langle a \rangle \ldots \langle a \rangle \langle b \rangle \ldots \langle b \rangle t$$

are true at s.

Describing Labeled Graphs

One last example:

- Suppose there are different rooms, painted either red or black, and there is a “transporter” in each room (“Beam me up Scotty”).

Describing Labeled Graphs

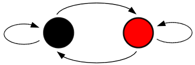
One last example:

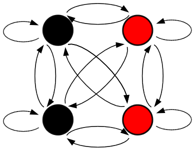
- Suppose there are different rooms, painted either red or black, and there is a “transporter” in each room (“Beam me up Scotty”).
- Each time we enter the transporter we are moved to a random room, accessible using the arrows.

Describing Labeled Graphs

One last example:

- Suppose there are different rooms, painted either red or black, and there is a "transporter" in each room ("Beam me up Scotty").
- Each time we enter the transporter we are moved to a random room, accessible using the arrows.
- Can we distinguish between the following two set ups?

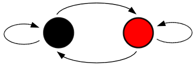


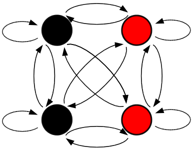


Describing Labeled Graphs

One last example:

- Suppose there are different rooms, painted either red or black, and there is a "transporter" in each room ("Beam me up Scotty").
- Each time we enter the transporter we are moved to a random room, accessible using the arrows.
- Can we distinguish between the following two set ups?





- FOL (with equality) can easily distinguish between these two structures (how?). The basic modal language does not.

The Basic Modal Language: Syntax and Semantics

- The syntax of the basic modal language is defined in terms of a signature defined by two infinite, countable, disjoint sets:
  - $\mathrm{PROP} = \{p_1, p_2, \ldots\}$, the set of *propositional variables*.
  - $\mathrm{MOD} = \{m_1, m_2, \ldots\}$, the set of *modalities*.

The Basic Modal Language: Syntax and Semantics

- The syntax of the basic modal language is defined in terms of a signature defined by two infinite, countable, disjoint sets:
  - $\text{PROP} = \{p_1, p_2, \ldots\}$, the set of *propositional variables*.
  - $\text{MOD} = \{m_1, m_2, \ldots\}$, the set of *modalities*.
- The set of formulas FORM in the signature $\langle \text{PROP}, \text{REL} \rangle$ is defined as:

$$\text{FORM} := p \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle m \rangle \varphi$$

where $p \in \text{PROP}, m \in \text{MOD}$ y $\varphi, \psi \in \text{FORM}$.

The Basic Modal Language: Syntax and Semantics

- The syntax of the basic modal language is defined in terms of a signature defined by two infinite, countable, disjoint sets:
  - $\text{PROP} = \{p_1, p_2, \ldots\}$, the set of *propositional variables*.
  - $\text{MOD} = \{m_1, m_2, \ldots\}$, the set of *modalities*.
- The set of formulas FORM in the signature $\langle \text{PROP}, \text{REL} \rangle$ is defined as:

$$\text{FORM} := p \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle m \rangle \varphi$$

  where $p \in \text{PROP}$, $m \in \text{MOD}$ y $\varphi, \psi \in \text{FORM}$.
- The operator $[m]$ is defined as $[m]\varphi = \neg\langle m \rangle\neg\varphi$ (similarly as how $\forall x.\varphi$ is defined as $\neg\exists x.\neg\varphi$).

The Basic Modal Language: Syntax and Semantics

To define the semantics, we introduce first Kripke Models.

The Basic Modal Language: Syntax and Semantics

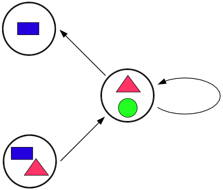
To define the semantics, we introduce first Kripke Models.

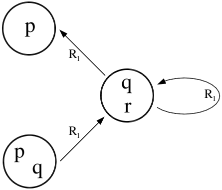
- A Kripke model is a structure $\mathcal{M} = \langle W, \{R_m\}, V \rangle$ where
  - W is a non-empy set of elements (the domain)
  - $\{R_m\}$ is a set of binary accessibility relations over W, one for each $m \in \textsc{mod}$.
  - $V : \textsc{prop} \to \wp(W)$ is a valuation function ($V(p)$ is the set of element where p holds).

The Basic Modal Language: Syntax and Semantics

To define the semantics, we introduce first Kripke Models.

- A Kripke model is a structure $\mathcal{M} = \langle W, \{R_m\}, V\rangle$ where
  - W is a non-empty set of elements (the domain)
  - $\{R_m\}$ is a set of binary accessibility relations over W, one for each $m \in \text{MOD}$.
  - $V : \text{PROP} \to \wp(W)$ is a valuation function ($V(p)$ is the set of element where p holds).
- Notice that $\mathcal{M}$ is a labeled directed graph or, from a more classical perspective, a relational structure.





The Basic Modal Language: Syntax and Semantics

Now we can define the semantics:

- Given a model $\mathcal{M} = \langle W, \{R_i\}, V \rangle$ and a state $w \in W$, the satisfaction relation is defined as:

The Basic Modal Language: Syntax and Semantics

Now we can define the semantics:

- Given a model $\mathcal{M} = \langle W, \{R_i\}, V \rangle$ and a state $w \in W$, the satisfaction relation is defined as:

  $\mathcal{M}, w \models p$ iff

The Basic Modal Language: Syntax and Semantics

Now we can define the semantics:

- Given a model $\mathcal{M} = \langle W, \{R_i\}, V\rangle$ and a state $w \in W$, the satisfaction relation is defined as:

  $$\mathcal{M}, w \models p \qquad \text{iff} \quad w \in V(p), p \in \text{PROP}$$

The Basic Modal Language: Syntax and Semantics

Now we can define the semantics:

- Given a model $\mathcal{M} = \langle W, \{R_i\}, V \rangle$ and a state $w \in W$, the satisfaction relation is defined as:

$$\begin{array}{lll} \mathcal{M}, w \models p & \text{iff} & w \in V(p), p \in \textsc{prop} \\ \mathcal{M}, w \models \neg\varphi & \text{iff} & \end{array}$$

The Basic Modal Language: Syntax and Semantics

Now we can define the semantics:

- Given a model $\mathcal{M} = \langle W, \{R_i\}, V\rangle$ and a state $w \in W$, the satisfaction relation is defined as:

$$
\begin{array}{lll}
\mathcal{M}, w \models p & \text{iff} & w \in V(p), p \in \text{PROP} \\
\mathcal{M}, w \models \neg\varphi & \text{iff} & \mathcal{M}, w \not\models \varphi
\end{array}
$$

The Basic Modal Language: Syntax and Semantics

Now we can define the semantics:

- Given a model $\mathcal{M} = \langle W, \{R_i\}, V\rangle$ and a state $w \in W$, the satisfaction relation is defined as:

$$
\begin{array}{lll}
\mathcal{M}, w \models p & \text{iff} & w \in V(p), p \in \text{PROP} \\
\mathcal{M}, w \models \neg\varphi & \text{iff} & \mathcal{M}, w \not\models \varphi \\
\mathcal{M}, w \models \varphi \wedge \psi & \text{iff} &
\end{array}
$$

The Basic Modal Language: Syntax and Semantics

Now we can define the semantics:

- Given a model $\mathcal{M} = \langle W, \{R_i\}, V\rangle$ and a state $w \in W$, the satisfaction relation is defined as:

$$
\begin{array}{lll}
\mathcal{M}, w \models p & \text{iff} & w \in V(p), p \in \text{PROP} \\
\mathcal{M}, w \models \neg\varphi & \text{iff} & \mathcal{M}, w \not\models \varphi \\
\mathcal{M}, w \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi
\end{array}
$$

The Basic Modal Language: Syntax and Semantics

Now we can define the semantics:

- Given a model $\mathcal{M} = \langle W, \{R_i\}, V\rangle$ and a state $w \in W$, the satisfaction relation is defined as:

$$
\begin{array}{lll}
\mathcal{M}, w \models p & \text{iff} & w \in V(p), p \in \text{PROP} \\
\mathcal{M}, w \models \neg\varphi & \text{iff} & \mathcal{M}, w \not\models \varphi \\
\mathcal{M}, w \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \langle R\rangle\varphi & \text{iff} &
\end{array}
$$

The Basic Modal Language: Syntax and Semantics

Now we can define the semantics:

- Given a model $\mathcal{M} = \langle W, \{R_i\}, V\rangle$ and a state $w \in W$, the satisfaction relation is defined as:

$$
\begin{array}{lll}
\mathcal{M}, w \models p & \text{iff} & w \in V(p),\, p \in \textsc{prop} \\
\mathcal{M}, w \models \neg\varphi & \text{iff} & \mathcal{M}, w \not\models \varphi \\
\mathcal{M}, w \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \langle R\rangle\varphi & \text{iff} & \exists w' \in W \text{ s.t. } R(w, w') \text{ and } \mathcal{M}, w' \models \varphi
\end{array}
$$

The Basic Modal Language: Syntax and Semantics

Now we can define the semantics:

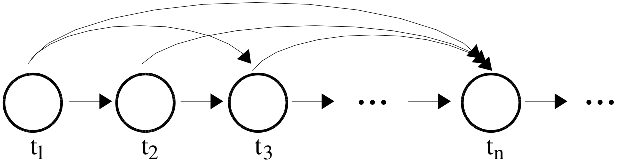
- Given a model $\mathcal{M} = \langle W, \{R_i\}, V \rangle$ and a state $w \in W$, the satisfaction relation is defined as:

$$
\begin{array}{lll}
\mathcal{M}, w \models p & \text{iff} & w \in V(p), p \in \text{PROP} \\
\mathcal{M}, w \models \neg\varphi & \text{iff} & \mathcal{M}, w \not\models \varphi \\
\mathcal{M}, w \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \langle R \rangle \varphi & \text{iff} & \exists w' \in W \text{ s.t. } R(w, w') \text{ and } \mathcal{M}, w' \models \varphi
\end{array}
$$

- We say that φ is valid in $\mathcal{M}$ iff for all $w \in W$ $\mathcal{M}, w \models \varphi$, and we write $\mathcal{M} \models \varphi$.

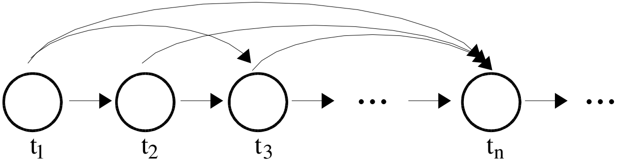
Extensions: Inverse

- Think about the following temporal line



Extensions: Inverse

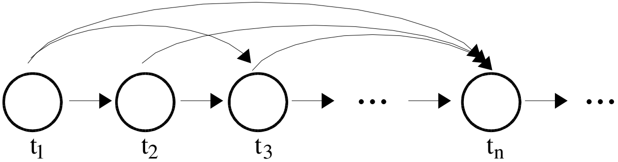
- Think about the following temporal line



- Clearly this structure can be seen as a Kripke model.

Extensions: Inverse

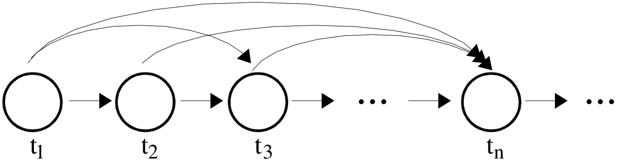
- Think about the following temporal line



- Clearly this structure can be seen as a Kripke model.
- The $\langle R \rangle$ operator means in some point in the future.

Extensions: Inverse

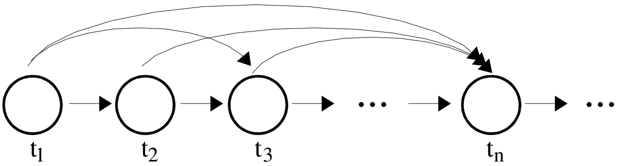
- Think about the following temporal line



- Clearly this structure can be seen as a Kripke model.
- The $\langle R \rangle$ operator means in some point in the future.
- The $[R]$ operator means in every point in the future.

Extensions: Inverse

- Think about the following temporal line



- Clearly this structure can be seen as a Kripke model.
- The $\langle R \rangle$ operator means in some point in the future.
- The $[R]$ operator means in every point in the future.
- Can we say, in this language, in some point in the past?

Extensions: Inverse

- We can introduce the inverse operator, written as $\langle R \rangle^{-1}$.

Extensions: Inverse

- We can introduce the inverse operator, written as $\langle R \rangle^{-1}$.
- We first extend the syntax:

$$\text{FORM} := p \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle m \rangle \varphi \mid \langle m \rangle^{-1} \varphi$$

Extensions: Inverse

- We can introduce the inverse operator, written as $\langle R \rangle^{-1}$.
- We first extend the syntax:

$$\text{FORM} := p \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle m \rangle \varphi \mid \langle m \rangle^{-1} \varphi$$

- Do we need to change our models?

Extensions: Inverse

- We extend the satisfaction relation. We had:

$$\begin{array}{lll}
\mathcal{M}, w \models p & \text{iff} & w \in V(p), p \in \text{PROP} \\
\mathcal{M}, w \models \neg\varphi & \text{iff} & \mathcal{M}, w \not\models \varphi \\
\mathcal{M}, w \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \langle R \rangle \varphi & \text{iff} & \exists w' \in W \text{ s.t. } R(w, w') \text{ and } \mathcal{M}, w' \models \varphi
\end{array}$$

Extensions: Inverse

- We extend the satisfaction relation. We had:

$$\begin{array}{lll} \mathcal{M}, w \models p & \text{iff} & w \in V(p), p \in \text{PROP} \\ \mathcal{M}, w \models \neg\varphi & \text{iff} & \mathcal{M}, w \not\models \varphi \\ \mathcal{M}, w \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi \\ \mathcal{M}, w \models \langle R\rangle\varphi & \text{iff} & \exists w' \in W \text{ s.t. } R(w, w') \text{ and } \mathcal{M}, w' \models \varphi \end{array}$$

- How can we define the semantics of the new operator?

$$\mathcal{M}, w \models \langle R\rangle^{-1}\varphi \quad \text{iff} \quad \exists w' \in W \text{ s.t. } R(w', w) \text{ and } \mathcal{M}, w' \models \varphi$$

Extensions: Universal Modality

- Another feature we might want is the possibility to describe a property that should be true in the whole model.

Extensions: Universal Modality

- Another feature we might want is the possibility to describe a property that should be true in the whole model.
- Consider the following situation. We are modeling a zoo, and we are interested in how bears are fed, and their relation with bear's keepers.

Extensions: Universal Modality

- Another feature we might want is the possibility to describe a property that should be true in the whole model.
- Consider the following situation. We are modeling a zoo, and we are interested in how bears are fed, and their relation with bear's keepers.
- We would like, for example, the following properties to be true in the model.

$$\begin{array}{cc} \mathit{bear} \lor \mathit{human} & \mathit{bear} \rightarrow [\textsc{child-of}]\mathit{bear} \\ \mathit{bear} \rightarrow \neg\mathit{human} & \mathit{bear} \rightarrow [\textsc{fed-by}](\mathit{human} \lor \mathit{bear}) \end{array}$$

Extensions: Universal Modality

- Another feature we might want is the possibility to describe a property that should be true in the whole model.
- Consider the following situation. We are modeling a zoo, and we are interested in how bears are fed, and their relation with bear's keepers.
- We would like, for example, the following properties to be true in the model.

$$\begin{array}{cc} bear \vee human & bear \rightarrow [\textsc{child-of}]bear \\ bear \rightarrow \neg human & bear \rightarrow [\textsc{fed-by}](human \vee bear) \end{array}$$

- Suppose we want to check whether the following situations are possible:

Extensions: Universal Modality

- Another feature we might want is the possibility to describe a property that should be true in the whole model.
- Consider the following situation. We are modeling a zoo, and we are interested in how bears are fed, and their relation with bear's keepers.
- We would like, for example, the following properties to be true in the model.

$$\begin{array}{cc} \mathit{bear} \vee \mathit{human} & \mathit{bear} \rightarrow [\text{CHILD-OF}]\mathit{bear} \\ \mathit{bear} \rightarrow \neg\mathit{human} & \mathit{bear} \rightarrow [\text{FED-BY}](\mathit{human} \vee \mathit{bear}) \end{array}$$

- Suppose we want to check whether the following situations are possible:
  - $\mathit{bear} \wedge \langle\text{FED-BY}\rangle(\neg\mathit{bear} \wedge \neg\mathit{human})$

Extensions: Universal Modality

- Another feature we might want is the possibility to describe a property that should be true in the whole model.
- Consider the following situation. We are modeling a zoo, and we are interested in how bears are fed, and their relation with bear's keepers.
- We would like, for example, the following properties to be true in the model.

$$\begin{array}{cc} \mathit{bear} \vee \mathit{human} & \mathit{bear} \rightarrow [\text{CHILD-OF}]\mathit{bear} \\ \mathit{bear} \rightarrow \neg\mathit{human} & \mathit{bear} \rightarrow [\text{FED-BY}](\mathit{human} \vee \mathit{bear}) \end{array}$$

- Suppose we want to check whether the following situations are possible:
  - $\mathit{bear} \wedge \langle\text{FED-BY}\rangle(\neg\mathit{bear} \wedge \neg\mathit{human})$
  - $\mathit{bear} \wedge \langle\text{CHILD-OF}\rangle(\mathit{bear} \wedge \mathit{human})$

Extensions: Universal Modality

- To be able to claim that a formula is true in the whole model, we can introduce the universal modality, notation A.

Extensions: Universal Modality

- To be able to claim that a formula is true in the whole model, we can introduce the universal modality, notation A.
- $A\varphi$ means that φ is true in all states in the model.

Extensions: Universal Modality

- To be able to claim that a formula is true in the whole model, we can introduce the universal modality, notation A.
- $A\varphi$ means that φ is true in all states in the model.
- What is the formal semantics for A?

Extensions: Universal Modality

- To be able to claim that a formula is true in the whole model, we can introduce the universal modality, notation A.
- $A\varphi$ means that φ is true in all states in the model.
- What is the formal semantics for A?
- And a question to think about: is this operator the same as the $\forall$ operator in FOL?

Extensions: PDL (Propositional Dynamic Logic)

- Suppose we want to model the behavior of programs.

Extensions: PDL (Propositional Dynamic Logic)

- Suppose we want to model the behavior of programs.
- For each program π we introduce a modality $\langle\pi\rangle$ (there will be an infinite number of modalities).

Extensions: PDL (Propositional Dynamic Logic)

- Suppose we want to model the behavior of programs.
- For each program π we introduce a modality $\langle\pi\rangle$ (there will be an infinite number of modalities).
- The intended interpretation of $\langle\pi\rangle\varphi$ will be: 'some (terminating) execution of π from the current state ends in a state satisfying φ'.

Extensions: PDL (Propositional Dynamic Logic)

- Suppose we want to model the behavior of programs.
- For each program π we introduce a modality $\langle \pi \rangle$ (there will be an infinite number of modalities).
- The intended interpretation of $\langle \pi \rangle \varphi$ will be: 'some (terminating) execution of π from the current state ends in a state satisfying φ'.
- Now we can model the inductive structure of progrmas: they can be composed, iterated, etc., building up other programs.

Extensions: PDL (Propositional Dynamic Logic)

Start with atomic programs a, b, c, etc., that are used to build more complex programs using the following syntax:

Extensions: PDL (Propositional Dynamic Logic)

Start with atomic programs a, b, c, etc., that are used to build more complex programs using the following syntax:

- (union) If π_1 y π_2 are programs, then $\pi_1 \cup \pi_2$ is a program that represents the non-deterministic execution of π_1 or π_2.

Extensions: PDL (Propositional Dynamic Logic)

Start with atomic programs a, b, c, etc., that are used to build more complex programs using the following syntax:

- (union) If π_1 y π_2 are programs, then $\pi_1 \cup \pi_2$ is a program that represents the non-deterministic execution of π_1 or π_2.
- (composition) If π_1 y π_2 are programs, then $\pi_1 ; \pi_2$ is a program obtained by first executing π_1 and then π_2.

Extensions: PDL (Propositional Dynamic Logic)

Start with atomic programs a, b, c, etc., that are used to build more complex programs using the following syntax:

- (union) If π_1 y π_2 are programs, then $\pi_1 \cup \pi_2$ is a program that represents the non-deterministic execution of π_1 or π_2.
- (composition) If π_1 y π_2 are programs, then $\pi_1; \pi_2$ is a program obtained by first executing π_1 and then π_2.
- (interaction) If π is a program then π^* is the program obtained by executing π a finite (including zero) number of times.

Extensions: PDL (Propositional Dynamic Logic)

Start with atomic programs a, b, c, etc., that are used to build more complex programs using the following syntax:

- (union) If π_1 y π_2 are programs, then $\pi_1 \cup \pi_2$ is a program that represents the non-deterministic execution of π_1 or π_2.
- (composition) If π_1 y π_2 are programs, then $\pi_1; \pi_2$ is a program obtained by first executing π_1 and then π_2.
- (interaction) If π is a program then π^* is the program obtained by executing π a finite (including zero) number of times.

E.g. if $\langle \pi_1 \rangle$ and $\langle \pi_2 \rangle$ are modal operators, then $\langle \pi_1 \cup \pi_2 \rangle$, $\langle \pi_1; \pi_2 \rangle$ and $\langle \pi^* \rangle$ are modal operators.

Extensions: PDL (Propositional Dynamic Logic)

- Modalities should be interpreted adequately so that they have the intended meaning.

Extensions: PDL (Propositional Dynamic Logic)

- Modalities should be interpreted adequately so that they have the intended meaning.
- Define

$$\begin{array}{rcl} R_{\pi_1 \cup \pi_2} & = & R_{\pi_1} \cup R_{\pi_2} \\ R_{\pi_1 ; \pi_2} & = & R_{\pi_1} \circ R_{\pi_2} \text{ (relation composition)} \\ R_{\pi^*} & = & (R_{\pi_1})^* \text{ (reflexive transitive closure of } R_{\pi_1}\text{)} \end{array}$$

Extensions: PDL (Propositional Dynamic Logic)

- Modalities should be interpreted adequately so that they have the intended meaning.
- Define

$$\begin{array}{rcl} R_{\pi_1 \cup \pi_2} & = & R_{\pi_1} \cup R_{\pi_2} \\ R_{\pi_1 ; \pi_2} & = & R_{\pi_1} \circ R_{\pi_2} \text{ (relation composition)} \\ R_{\pi^*} & = & (R_{\pi_1})^* \text{ (reflexive transitive closure of } R_{\pi_1}) \end{array}$$

- Then,

$$\mathcal{M}, w \models \langle \pi \rangle \varphi \text{ iff } \exists w'. R_\pi(w, w') \text{ y } \mathcal{M}, w' \models \varphi.$$

Extensions: PDL (Propositional Dynamic Logic)

- With this definitions, what is the meaning of the following formula?

$$\langle \pi^* \rangle \varphi \leftrightarrow \varphi \vee \langle \pi ; \pi^* \rangle \varphi$$

Extensions: PDL (Propositional Dynamic Logic)

- With this definitions, what is the meaning of the following formula?

$$\langle \pi^* \rangle \varphi \leftrightarrow \varphi \vee \langle \pi ; \pi^* \rangle \varphi$$

- Should it be valid?

Extensions: PDL (Propositional Dynamic Logic)

- With this definitions, what is the meaning of the following formula?

$$\langle \pi^* \rangle \varphi \leftrightarrow \varphi \vee \langle \pi; \pi^* \rangle \varphi$$

- Should it be valid? Prove it!

Extensions: PDL (Propositional Dynamic Logic)

- With this definitions, what is the meaning of the following formula?

$$\langle \pi^* \rangle \varphi \leftrightarrow \varphi \vee \langle \pi; \pi^* \rangle \varphi$$

- Should it be valid? Prove it!
- And this?

$$[\pi^*](\varphi \to [\pi]\varphi) \to (\varphi \to [\pi^*]\varphi)$$

Extensions: PDL (Propositional Dynamic Logic)

- With this definitions, what is the meaning of the following formula?

$$\langle \pi^* \rangle \varphi \leftrightarrow \varphi \vee \langle \pi ; \pi^* \rangle \varphi$$

- Should it be valid? Prove it!
- And this?

$$[\pi^*](\varphi \rightarrow [\pi]\varphi) \rightarrow (\varphi \rightarrow [\pi^*]\varphi)$$

- Think about induction...

Tests

Sometimes, it is useful to include tests as programs.

- Syntax: If φ is a PDL formula, then $\varphi?$ is a PDL program.
- Semantics: $R_{\varphi?} = \{(w, w) \mid \mathcal{M}, w \models \varphi\}$

Notice how we can now "transform" formulas into programs. What do you think of the following formula

$$\langle p? \rangle q \leftrightarrow p \wedge q?$$

Tests

Sometimes, it is useful to include tests as programs.

- Syntax: If φ is a PDL formula, then $\varphi?$ is a PDL program.
- Semantics: $R_{\varphi?} = \{(w, w) \mid \mathcal{M}, w \models \varphi\}$

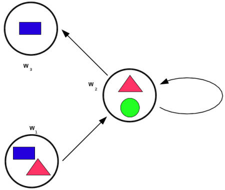
Notice how we can now "transform" formulas into programs. What do you think of the following formula

$$\langle p? \rangle q \leftrightarrow p \wedge q?$$

Are they useful then?

Extensions: Hybrid Logics

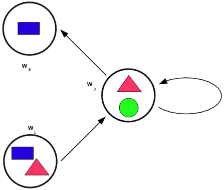
Let us go back to simple labeled graphs.



- Can w_1 see itself?
- Are w_1 and w_2 two different states?

Extensions: Hybrid Logics

Let us go back to simple labeled graphs.



- Can w_1 see itself?
- Are w_1 and w_2 two different states?

- The basic modal language does not have constants or equality.
  We can add means to name and compare states.

Extensions: Hybrid Logics

$\mathcal{HL}(@)$ is the extension of the basic modal logic with

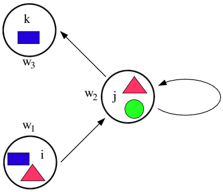
- names (nominals): they are a news set of atomic symbols, used to represent states. The key is to ensure that a nominal is true at a single state in a model. In general we will write them as $i, j, k, \ldots$
- @: $@_i\varphi$ holds iff φ is true in the state named by i.

Extensions: Hybrid Logics

$\mathcal{HL}(@)$ is the extension of the basic modal logic with

- names (nominals): they are a news set of atomic symbols, used to represent states. The key is to ensure that a nominal is true at a single state in a model. In general we will write them as $i, j, k, \ldots$
- @: $@_i\varphi$ holds iff φ is true in the state named by i.

Now we can express …



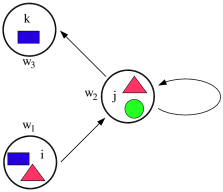
- Can w_1 see itself?
  $@_i\langle see\rangle i$

Extensions: Hybrid Logics

$\mathcal{HL}(@)$ is the extension of the basic modal logic with

- names (nominals): they are a news set of atomic symbols, used to represent states. The key is to ensure that a nominal is true at a single state in a model. In general we will write them as $i, j, k, \ldots$
- @: $@_i\varphi$ holds iff φ is true in the state named by i.

Now we can express ...



- Can w_1 see itself?
  $@_i\langle see\rangle i$
- Are w_1 and w_2 the same state?
  $@_i\neg j$

Extensions: Hybrid Logics

So, the syntax is define as:

- To a signature $\langle \text{PROP}, \text{MOD} \rangle$ we had before, we ad a new set $\text{NOM} = \{i, j, k, \dots\}$ of nominals.

Extensions: Hybrid Logics

So, the syntax is define as:

- To a signature $\langle \text{PROP}, \text{MOD} \rangle$ we had before, we ad a new set $\text{NOM} = \{i, j, k, \ldots\}$ of nominals.
- Well formed formulas are defined as

$$\text{FORM} := p \mid i \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle m \rangle \varphi \mid @_i \varphi$$

where $p \in \text{PROP}$, $m \in \text{MOD}$, $i \in \text{NOM}$ y $\varphi, \psi \in \text{FORM}$.

Extensions: Hybrid Logics

So, the syntax is define as:

- To a signature $\langle \text{PROP}, \text{MOD} \rangle$ we had before, we ad a new set $\text{NOM} = \{i, j, k, \dots\}$ of nominals.
- Well formed formulas are defined as

$$\text{FORM} := p \mid i \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle m \rangle \varphi \mid @_i \varphi$$

  where $p \in \text{PROP}, m \in \text{MOD}, i \in \text{NOM}$ y $\varphi, \psi \in \text{FORM}$.
- Do we need to do any change in the models? How is the semantics of $\mathcal{HL}(@)$ defined?

Extensions: Hybrid Logics

So, the syntax is define as:

- To a signature $\langle \text{PROP}, \text{MOD} \rangle$ we had before, we ad a new set $\text{NOM} = \{i, j, k, \ldots\}$ of nominals.
- Well formed formulas are defined as

$$\text{FORM} := p \mid i \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle m \rangle \varphi \mid @_i \varphi$$

  where $p \in \text{PROP}$, $m \in \text{MOD}$, $i \in \text{NOM}$ y $\varphi, \psi \in \text{FORM}$.
- Do we need to do any change in the models? How is the semantics of $\mathcal{HL}(@)$ defined?
- Exercise!