LogicS

Lecture #6: No Way to Say Warm in French

Carlos Areces and Patrick Blackburn

{carlos.areces,patrick.blackburn}@loria.fr

INRIA Nancy Grand Est
Nancy, France

NASSLLI 2010 - Bloomington - USA

The Story so Far

The Story so Far

- We are working with the $\langle R \rangle$ language

The Story so Far

- We are working with the $\langle R \rangle$ language
- We saw that the lenguage cannot say
  - "I am not a tree"
  - "I am infinite"

The Story so Far

- We are working with the $\langle R \rangle$ language
- We saw that the lenguage cannot say
  - "I am not a tree"
  - "I am infinite"
- On the positive side
  - It has a simple reasoning calculus: labelled tableaux
  - It is decidable.

What do we do Today

What do we do Today

- We will extend the expressive power of the language. . .
- . . . and explore quite a number of possibilities.

What do we do Today

- We will extend the expressive power of the language...
- ...and explore quite a number of possibilities.
- We will learn to count up to n.

What do we do Today

- We will extend the expressive power of the language...
- ...and explore quite a number of possibilities.
- We will learn to count up to n.
- We will learn to name nodes.

What do we do Today

- We will extend the expressive power of the language...
- ...and explore quite a number of possibilities.
- We will learn to count up to n.
- We will learn to name nodes.
- We will learn to say "everywhere".

What do we do Today

- We will extend the expressive power of the language...
- ...and explore quite a number of possibilities.
- We will learn to count up to n.
- We will learn to name nodes.
- We will learn to say "everywhere".
- We will learn to say "it won't take forever".

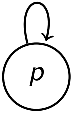
One is the Same as Infinite

One is the Same as Infinite

- We said in the previos lecture that we cannot say infinite in the $\langle R \rangle$ language.

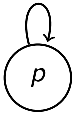
One is the Same as Infinite

- We said in the previos lecture that we cannot say infinite in the $\langle R \rangle$ language.
- Let's see this in more detail. Consider the model:



One is the Same as Infinite

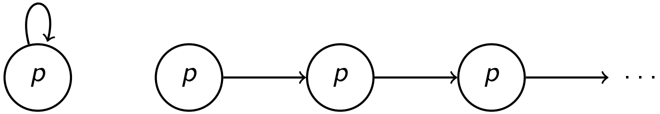
- We said in the previos lecture that we cannot say infinite in the $\langle R \rangle$ language.
- Let's see this in more detail. Consider the model:



- This is not a tree. Hence, there should be a tree like structure which should be the same as this one for the $\langle R \rangle$ language.

One is the Same as Infinite

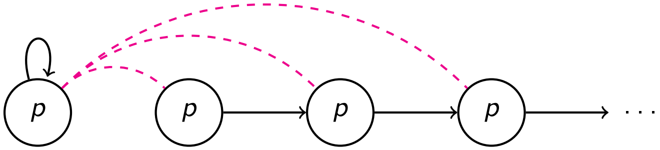
- We said in the previos lecture that we cannot say infinite in the $\langle R \rangle$ language.
- Let's see this in more detail. Consider the model:



- This is not a tree. Hence, there should be a tree like structure which should be the same as this one for the $\langle R \rangle$ language.

One is the Same as Infinite

- We said in the previos lecture that we cannot say infinite in the $\langle R \rangle$ language.
- Let's see this in more detail. Consider the model:



- This is not a tree. Hence, there should be a tree like structure which should be the same as this one for the $\langle R \rangle$ language.

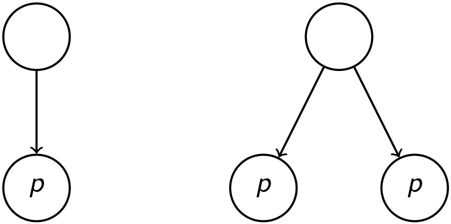
One is the Same as Two

One is the Same as Two

- But let's consider a simpler example (after all, infinite is quite a big number).

One is the Same as Two

- But let's consider a simpler example (after all, infinite is quite a big number).
- We saw that the $\langle R \rangle$ language cannot distinguish between one and two!!!



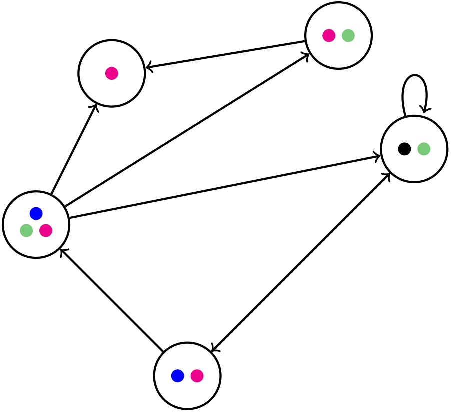
Learning to Count

Learning to Count

Suppose we want to say that two green nodes are accessible . . .

Learning to Count

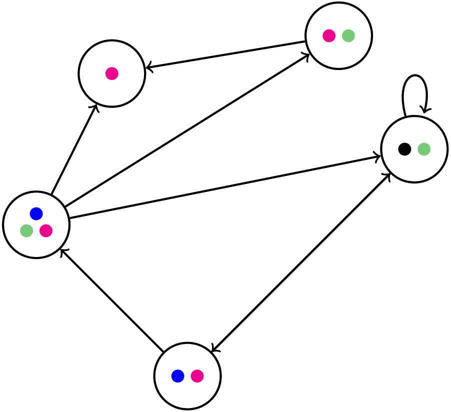
Suppose we want to say that two green nodes are accessible . . .



Learning to Count

Suppose we want to say that two green nodes are accessible . . .

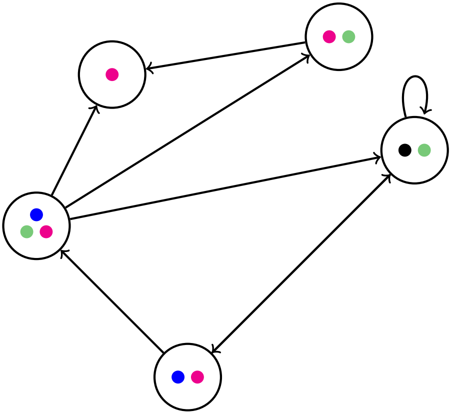
$\langle R \rangle$green $\wedge$ $\langle R \rangle$green



Learning to Count

Suppose we want to say that two green nodes are accessible . . .

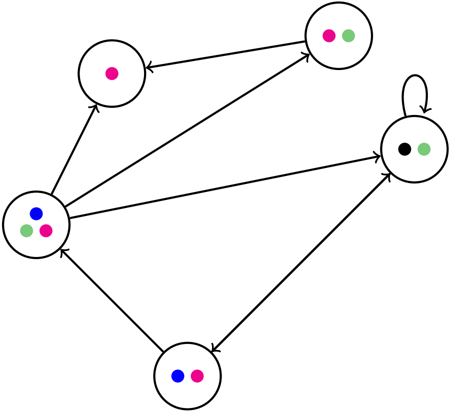
$\langle R \rangle$(green $\wedge$ pink) $\wedge$
$\langle R \rangle$(green $\wedge \neg$ pink)



Learning to Count

Suppose we want to say that two green nodes are accessible ...

$\langle = 2\ R \rangle$ green



Alice in Wonderland

Alice in Wonderland

Humpty Dumpty: When I use a word, it means just what I choose it to mean – neither more nor less.
Alice: The question is, whether you can make words mean so many different things.
Humpty Dumpty: The question is: which is to be master – that's all.

Alice in Wonderland

> **Humpty Dumpty**: When I use a word, it means just what I choose it to mean – neither more nor less.
> **Alice**: The question is, whether you can make words mean so many different things.
> **Humpty Dumpty**: The question is: which is to be master – that's all.

- If the language cannot express something we are interested in, we just extend the language!

Alice in Wonderland

Humpty Dumpty: When I use a word, it means just what I choose it to mean – neither more nor less.
Alice: The question is, whether you can make words mean so many different things.
Humpty Dumpty: The question is: which is to be master – that's all.

- If the language cannot express something we are interested in, we just extend the language!
- Counting successors:

$$\mathcal{M}, w \models \langle = n\ R \rangle \varphi \text{ iff } |\{w' \mid wRw' \text{ and } \mathcal{M}, w' \models \varphi\}| = n$$

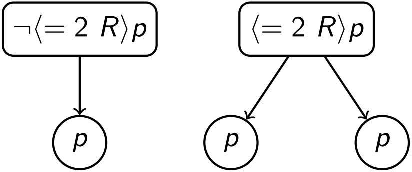
Alice in Wonderland

Humpty Dumpty: When I use a word, it means just what I choose it to mean – neither more nor less.
Alice: The question is, whether you can make words mean so many different things.
Humpty Dumpty: The question is: which is to be master – that's all.

- If the language cannot express something we are interested in, we just extend the language!
- Counting successors:

$$\mathcal{M}, w \models \langle = n\ R\rangle\varphi \text{ iff } |\{w' \mid wRw' \text{ and } \mathcal{M}, w' \models \varphi\}| = n$$

- Clearly:



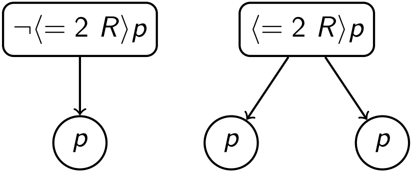
Alice in Wonderland

Humpty Dumpty: When I use a word, it means just what I choose it to mean – neither more nor less.
Alice: The question is, whether you can make words mean so many different things.
Humpty Dumpty: The question is: which is to be master – that's all.

- If the language cannot express something we are interested in, we just extend the language!
- Counting successors:

$$\mathcal{M}, w \models \langle = n\ R\rangle\varphi \text{ iff } |\{w' \mid wRw' \text{ and } \mathcal{M}, w' \models \varphi\}| = n$$

- Clearly:



The models are not the same for the $\langle = n\ R\rangle$ language.

Extending the Language

Extending the Language

- What other things we cannot say in the $\langle R \rangle$ language?

Extending the Language

- What other things we cannot say in the $\langle R \rangle$ language?
- Plenty:
  1. In that particular node.

Extending the Language

- What other things we cannot say in the $\langle R \rangle$ language?
- Plenty:
  1. In that particular node.
  2. Everywhere in the model.

Extending the Language

- What other things we cannot say in the $\langle R \rangle$ language?
- Plenty:
  1. In that particular node.
  2. Everywhere in the model.
  3. In a finite number of steps.

     ...

Extending the Language

- What other things we cannot say in the $\langle R \rangle$ language?
- Plenty:
  1. In that particular node.
  2. Everywhere in the model.
  3. In a finite number of steps.

     ...
- Luckily, as Humpty Dumpty says, we are the masters, and we can desing the language that better pleases us.

Extending the Language

- What other things we cannot say in the $\langle R \rangle$ language?
- Plenty:
  1. In that particular node.
  2. Everywhere in the model.
  3. In a finite number of steps.

     ...
- Luckily, as Humpty Dumpty says, we are the masters, and we can desing the language that better pleases us.
- Let's get to work...

Names for Points

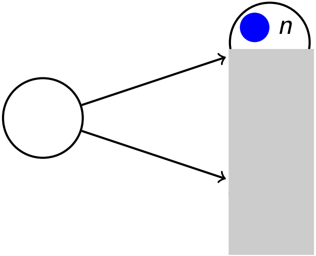
Names for Points

Suppose that n is a name for a point. That is, it can label a unique point in any relational structure.

Names for Points

Suppose that n is a name for a point. That is, it can label a unique point in any relational structure.

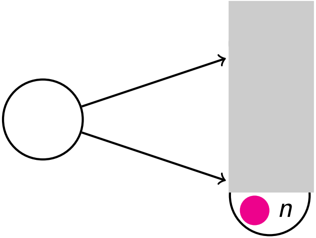
$\langle R \rangle(\mathrm{blue} \wedge n)$



Names for Points

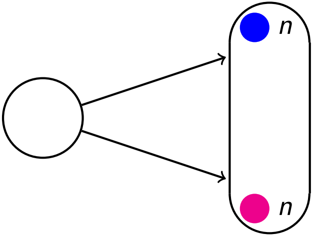
Suppose that n is a name for a point. That is, it can label a unique point in any relational structure.

$\langle R \rangle(\mathsf{pink} \wedge n)$



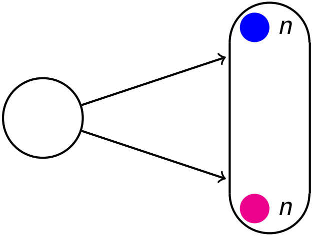
Names for Points

Suppose that n is a name for a point. That is, it can label a unique point in any relational structure.



Names for Points

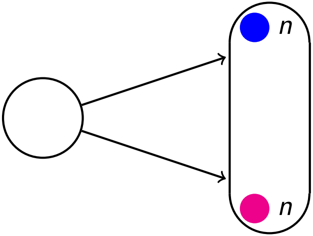
Suppose that n is a name for a point. That is, it can label a unique point in any relational structure.



- Looks useful...

Names for Points

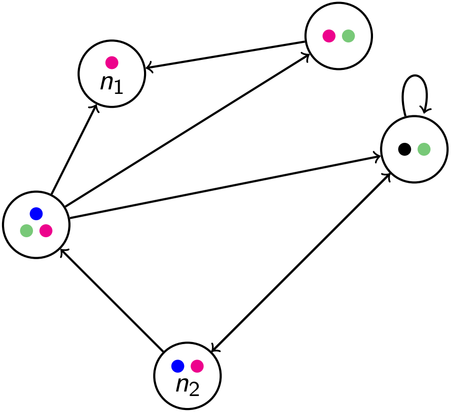
Suppose that n is a name for a point. That is, it can label a unique point in any relational structure.



- Looks useful...
- We can introduce names into our language (you probably know them as constants).

Names for Points

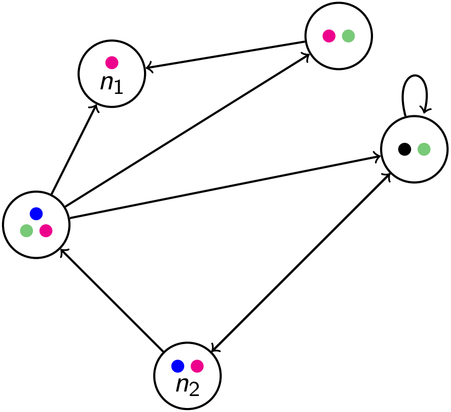
- When we allow names in our language, our models will look like this:



Names for Points

- When we allow names in our language, our models will look like this:

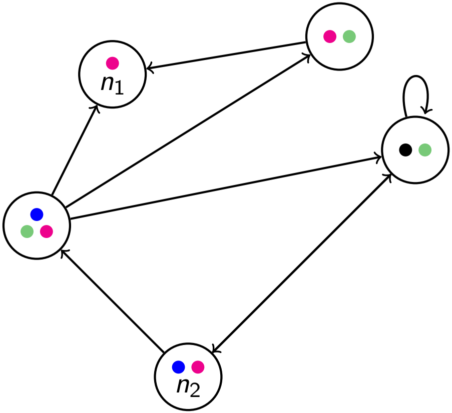
- We have already used something like names.



Names for Points

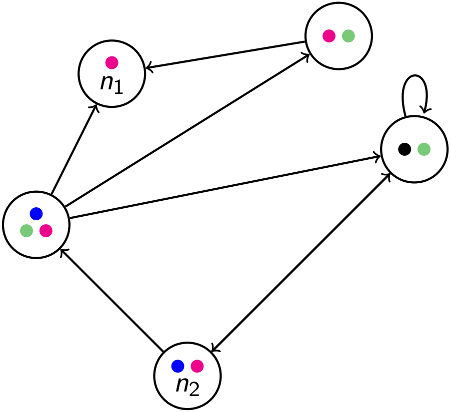
- When we allow names in our language, our models will look like this:

- We have already used something like names. Anybody remembers when?



Names for Points

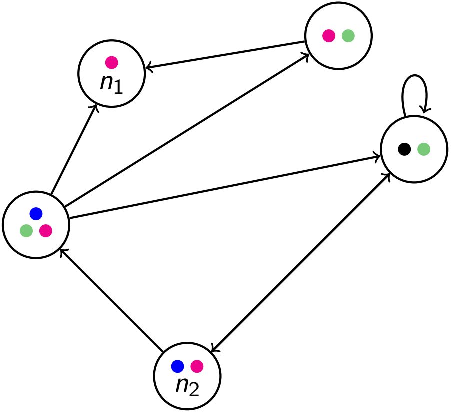
- When we allow names in our language, our models will look like this:



- We have already used something like names. Anybody remembers when?
- Tableaux for the $\langle R \rangle$ language!

Names for Points

- When we allow names in our language, our models will look like this:

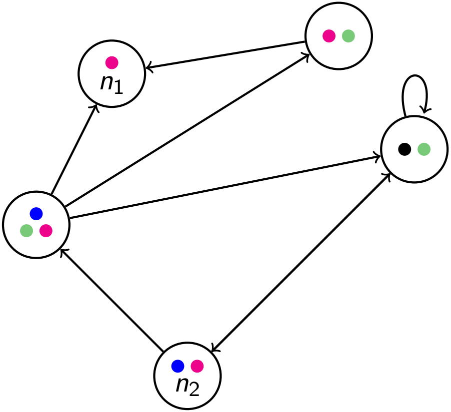


- We have already used something like names. Anybody remembers when?
- Tableaux for the $\langle R \rangle$ language!
- If we also introduce the :-operator we can write things like

$$n_1\text{:pink}$$

Names for Points

- When we allow names in our language, our models will look like this:

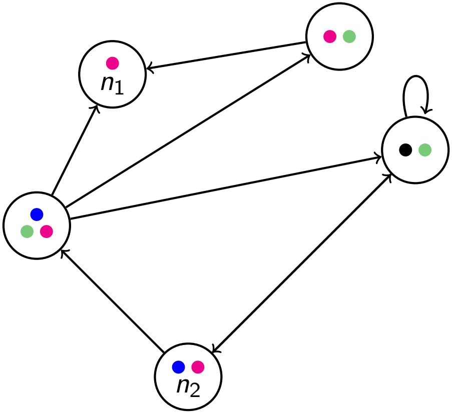


- We have already used something like names. Anybody remembers when?
- Tableaux for the $\langle R \rangle$ language!
- If we also introduce the :-operator we can write things like

$$n_1\text{:pink}$$
$$n_2\text{:}\langle R \rangle\text{pink}$$

Names for Points

- When we allow names in our language, our models will look like this:



- We have already used something like names. Anybody remembers when?
- Tableaux for the $\langle R \rangle$ language!
- If we also introduce the :-operator we can write things like

$$n_1\text{:pink}$$
$$n_2\text{:}\langle R \rangle\text{pink}$$
$$n_2\text{:}\langle R \rangle\langle R \rangle\langle R \rangle n_2$$

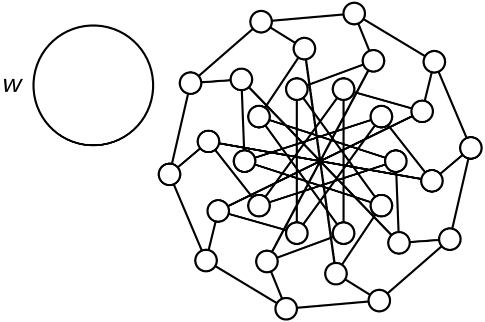
Everywhere in the model

Everywhere in the model

- Suppose we want to paint everything pink (we love pink).

Everywhere in the model

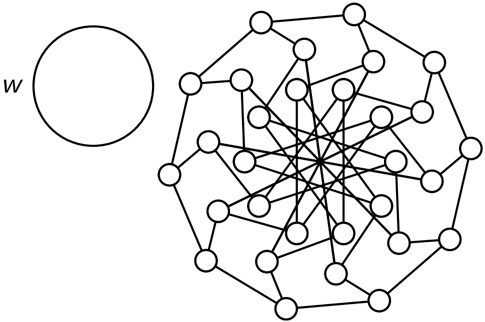
- Suppose we want to paint everything pink (we love pink).
- Can we do it? Let's see and example, consider this model:



Everywhere in the model

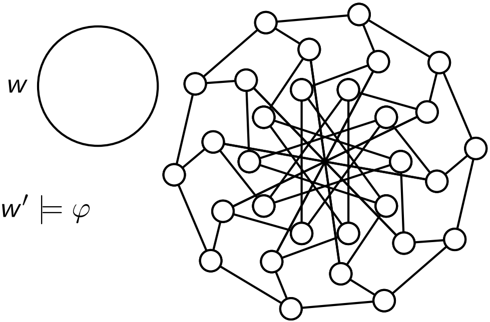
- Suppose we want to paint everything pink (we love pink).
- Can we do it? Let's see and example, consider this model:

- Is there a formula of the $\langle R \rangle$ language, that we can make true at w, so that pink is true everywhere in the model?



Everywhere in the model

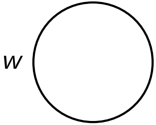
- Suppose we want to paint everything pink (we love pink).
- Can we do it? Let's see and example, consider this model:

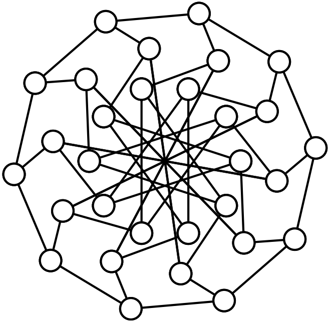


- Is there a formula of the $\langle R \rangle$ language, that we can make true at w, so that pink is true everywhere in the model?
- Define the $[U]$ operator as $\mathcal{M}, w \models [U]\varphi$ iff for all w', $\mathcal{M}, w' \models \varphi$

Everywhere in the model

- Suppose we want to paint everything pink (we love pink).
- Can we do it? Let's see and example, consider this model:



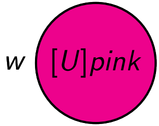


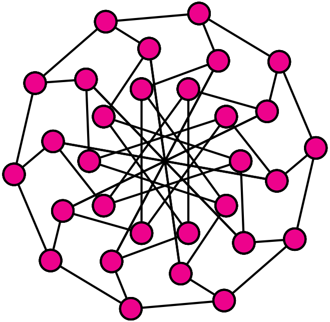
- Is there a formula of the $\langle R \rangle$ language, that we can make true at w, so that pink is true everywhere in the model?
- Define the $[U]$ operator as $\mathcal{M}, w \models [U]\varphi$ iff for all w', $\mathcal{M}, w' \models \varphi$
- Then $\mathcal{M}, w \models [U]\text{pink}$ if the whole model is pink.

Everywhere in the model

- Suppose we want to paint everything pink (we love pink).
- Can we do it? Let's see and example, consider this model:

- Is there a formula of the $\langle R \rangle$ language, that we can make true at w, so that pink is true everywhere in the model?
- Define the $[U]$ operator as $\mathcal{M}, w \models [U]\varphi$ iff for all w', $\mathcal{M}, w' \models \varphi$
- Then $\mathcal{M}, w \models [U]\text{pink}$ if the whole model is pink.

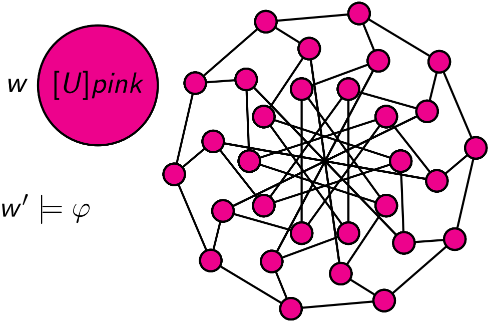




Everywhere in the model

- Suppose we want to paint everything pink (we love pink).
- Can we do it? Let's see and example, consider this model:

- Is there a formula of the $\langle R \rangle$ language, that we can make true at w, so that pink is true everywhere in the model?
- Define the $[U]$ operator as $\mathcal{M}, w \models [U]\varphi$ iff for all w', $\mathcal{M}, w' \models \varphi$
- Then $\mathcal{M}, w \models [U]$pink if the whole model is pink.
- Jah!



In a Finite Number of Steps

In a Finite Number of Steps

- In the $\langle R \rangle$ language we can say
  - In one step p: $\langle R \rangle p$

In a Finite Number of Steps

- In the $\langle R \rangle$ language we can say
  - In one step p: $\langle R \rangle p$
  - In two steps p: $\langle R \rangle \langle R \rangle p$

In a Finite Number of Steps

- In the $\langle R \rangle$ language we can say
  - In one step p: $\langle R \rangle p$
  - In two steps p: $\langle R \rangle \langle R \rangle p$
  - ...
  - But we cannot say, in a finite (zero or more, but unspecified) number of steps p:
    $$p \vee \langle R \rangle p \vee \langle R \rangle \langle R \rangle p \vee \ldots$$

In a Finite Number of Steps

- In the $\langle R \rangle$ language we can say
  - In one step p: $\langle R \rangle p$
  - In two steps p: $\langle R \rangle \langle R \rangle p$
  - ...
  - But we cannot say, in a finite (zero or more, but unspecified) number of steps p:
    $$p \vee \langle R \rangle p \vee \langle R \rangle \langle R \rangle p \vee \ldots$$
- Define the $\langle R^* \rangle$ operator as

  $\mathcal{M}, w \models \langle R^* \rangle \varphi$ iff there is w' s.t. wR^*w' and $\mathcal{M}, w' \models \varphi$

  for R^* is the reflexive and transitive closure of R.

In a Finite Number of Steps

- In the $\langle R \rangle$ language we can say
  - In one step p: $\langle R \rangle p$
  - In two steps p: $\langle R \rangle \langle R \rangle p$
  - ...
  - But we cannot say, in a finite (zero or more, but unspecified) number of steps p:
    $$p \vee \langle R \rangle p \vee \langle R \rangle \langle R \rangle p \vee \ldots$$
- Define the $\langle R^* \rangle$ operator as

  $\mathcal{M}, w \models \langle R^* \rangle \varphi$ iff there is w' s.t. wR^*w' and $\mathcal{M}, w' \models \varphi$

  for R^* is the reflexive and transitive closure of R.
  (Let's write $[R^*]\varphi$ for $\neg \langle R^* \rangle \neg \varphi$)

In a Finite Number of Steps

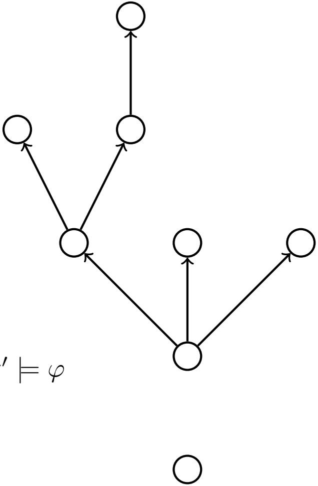
- In the $\langle R \rangle$ language we can say
  - In one step p: $\langle R \rangle p$
  - In two steps p: $\langle R \rangle \langle R \rangle p$
  - ...
  - But we cannot say, in a finite (zero or more, but unspecified) number of steps p:

$$p \vee \langle R \rangle p \vee \langle R \rangle \langle R \rangle p \vee \ldots$$

- Define the $\langle R^* \rangle$ operator as

$\mathcal{M}, w \models \langle R^* \rangle \varphi$ iff there is w' s.t. wR^*w' and $\mathcal{M}, w' \models \varphi$

for R^* is the reflexive and transitive closure of R.
(Let's write $[R^*]\varphi$ for $\neg\langle R^* \rangle \neg\varphi$)

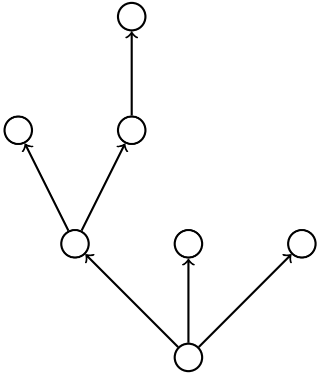


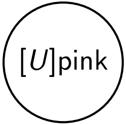
In a Finite Number of Steps

- In the $\langle R \rangle$ language we can say
  - In one step p: $\langle R \rangle p$
  - In two steps p: $\langle R \rangle \langle R \rangle p$
  - ...
  - But we cannot say, in a finite (zero or more, but unspecified) number of steps p:
    $$p \vee \langle R \rangle p \vee \langle R \rangle \langle R \rangle p \vee \ldots$$
- Define the $\langle R^* \rangle$ operator as

  $\mathcal{M}, w \models \langle R^* \rangle \varphi$ iff there is w' s.t. wR^*w' and $\mathcal{M}, w' \models \varphi$

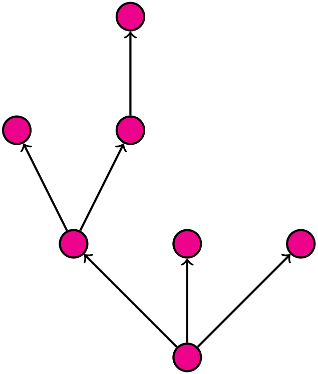
  for R^* is the reflexive and transitive closure of R.
  (Let's write $[R^*]\varphi$ for $\neg\langle R^* \rangle \neg\varphi$)





In a Finite Number of Steps

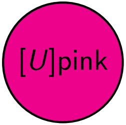
- In the $\langle R \rangle$ language we can say
  - In one step p: $\langle R \rangle p$
  - In two steps p: $\langle R \rangle \langle R \rangle p$
  - ...
  - But we cannot say, in a finite (zero or more, but unspecified) number of steps p:
    $$p \vee \langle R \rangle p \vee \langle R \rangle \langle R \rangle p \vee \ldots$$



- Define the $\langle R^* \rangle$ operator as

  $\mathcal{M}, w \models \langle R^* \rangle \varphi$ iff there is w' s.t. wR^*w' and $\mathcal{M}, w' \models \varphi$

  for R^* is the reflexive and transitive closure of R.
  (Let's write $[R^*]\varphi$ for $\neg \langle R^* \rangle \neg \varphi$)

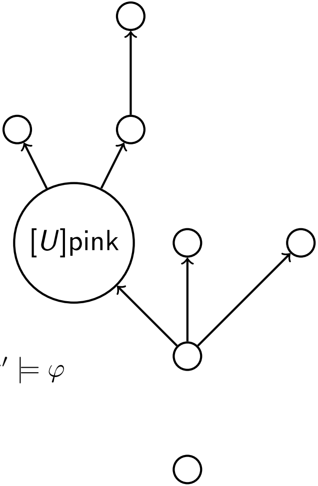


In a Finite Number of Steps

- In the $\langle R \rangle$ language we can say
  - In one step p: $\langle R \rangle p$
  - In two steps p: $\langle R \rangle \langle R \rangle p$
  - ...
  - But we cannot say, in a finite (zero or more, but unspecified) number of steps p:
    $$p \vee \langle R \rangle p \vee \langle R \rangle \langle R \rangle p \vee \dots$$
- Define the $\langle R^* \rangle$ operator as

  $\mathcal{M}, w \models \langle R^* \rangle \varphi$ iff there is w' s.t. wR^*w' and $\mathcal{M}, w' \models \varphi$

  for R^* is the reflexive and transitive closure of R.
  (Let's write $[R^*]\varphi$ for $\neg \langle R^* \rangle \neg \varphi$)

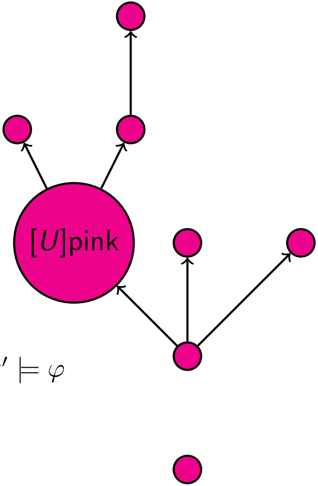


In a Finite Number of Steps

- In the $\langle R \rangle$ language we can say
  - In one step p: $\langle R \rangle p$
  - In two steps p: $\langle R \rangle \langle R \rangle p$
  - ...
  - But we cannot say, in a finite (zero or more, but unspecified) number of steps p:
    $$p \vee \langle R \rangle p \vee \langle R \rangle \langle R \rangle p \vee \dots$$
- Define the $\langle R^* \rangle$ operator as

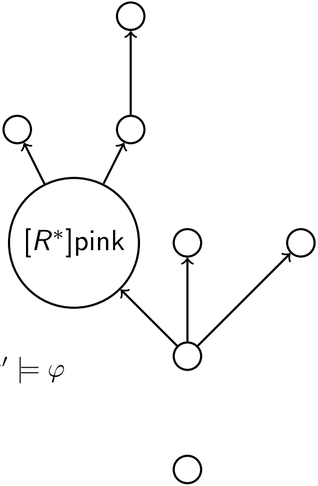
  $\mathcal{M}, w \models \langle R^* \rangle \varphi$ iff there is w' s.t. wR^*w' and $\mathcal{M}, w' \models \varphi$

  for R^* is the reflexive and transitive closure of R.
  (Let's write $[R^*]\varphi$ for $\neg \langle R^* \rangle \neg \varphi$)



In a Finite Number of Steps

- In the $\langle R \rangle$ language we can say
  - In one step p: $\langle R \rangle p$
  - In two steps p: $\langle R \rangle \langle R \rangle p$
  - ...
  - But we cannot say, in a finite (zero or more, but unspecified) number of steps p:
    $$p \vee \langle R \rangle p \vee \langle R \rangle \langle R \rangle p \vee \dots$$



- Define the $\langle R^* \rangle$ operator as

  $\mathcal{M}, w \models \langle R^* \rangle \varphi$ iff there is w' s.t. wR^*w' and $\mathcal{M}, w' \models \varphi$

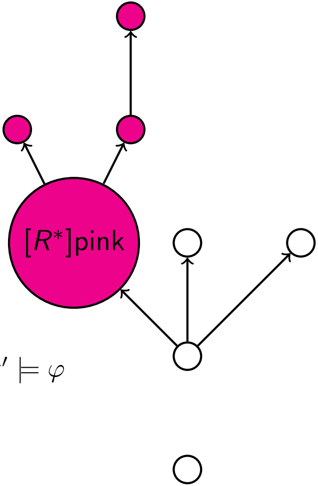
  for R^* is the reflexive and transitive closure of R.
  (Let's write $[R^*]\varphi$ for $\neg \langle R^* \rangle \neg \varphi$)

In a Finite Number of Steps

- In the $\langle R \rangle$ language we can say
  - In one step p: $\langle R \rangle p$
  - In two steps p: $\langle R \rangle \langle R \rangle p$
  - ...
  - But we cannot say, in a finite (zero or more, but unspecified) number of steps p:
    $$p \vee \langle R \rangle p \vee \langle R \rangle \langle R \rangle p \vee \ldots$$
- Define the $\langle R^* \rangle$ operator as

  $\mathcal{M}, w \models \langle R^* \rangle \varphi$ iff there is w' s.t. wR^*w' and $\mathcal{M}, w' \models \varphi$

  for R^* is the reflexive and transitive closure of R.
  (Let's write $[R^*]\varphi$ for $\neg \langle R^* \rangle \neg \varphi$)

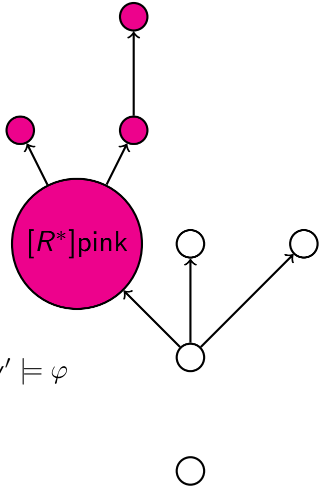


In a Finite Number of Steps

- In the $\langle R \rangle$ language we can say
  - In one step p: $\langle R \rangle p$
  - In two steps p: $\langle R \rangle \langle R \rangle p$
  - ...
  - But we cannot say, in a finite (zero or more, but unspecified) number of steps p:
    $$p \vee \langle R \rangle p \vee \langle R \rangle \langle R \rangle p \vee \ldots$$
- Define the $\langle R^* \rangle$ operator as

  $\mathcal{M}, w \models \langle R^* \rangle \varphi$ iff there is w' s.t. wR^*w' and $\mathcal{M}, w' \models \varphi$

  for R^* is the reflexive and transitive closure of R.
  (Let's write $[R^*]\varphi$ for $\neg \langle R^* \rangle \neg \varphi$)



The Complete Menu

The Complete Menu

Our models are structures like
$\mathcal{M} = \langle W, \{R_1, \ldots, R_n\}, \{P_1, \ldots, P_m\}, \{N_1, \ldots, N_k\} \rangle$

The Complete Menu

Our models are structures like
$\mathcal{M} = \langle W, \{R_1, \ldots, R_n\}, \{P_1, \ldots, P_m\}, \{N_1, \ldots, N_k\}\rangle$

- Counting successors:
  $\mathcal{M}, w \models \langle = n\ R\rangle\varphi$ iff $|\{w' \mid wRw' \text{ and } \mathcal{M}, w' \models \varphi\}| = n$.

The Complete Menu

Our models are structures like
$\mathcal{M} = \langle W, \{R_1, \dots, R_n\}, \{P_1, \dots, P_m\}, \{N_1, \dots, N_k\} \rangle$

- Counting successors:
  $\mathcal{M}, w \models \langle = n\ R \rangle \varphi$ iff $|\{w' \mid wRw'$ and $\mathcal{M}, w' \models \varphi\}| = n$.
- Names and the :-operator:

$$\mathcal{M}, w \models n_i \text{ iff } w = N_i$$

The Complete Menu

Our models are structures like
$\mathcal{M} = \langle W, \{R_1, \ldots, R_n\}, \{P_1, \ldots, P_m\}, \{N_1, \ldots, N_k\}\rangle$

- Counting successors:
  $\mathcal{M}, w \models \langle = n\ R\rangle\varphi$ iff $|\{w' \mid wRw'$ and $\mathcal{M}, w' \models \varphi\}| = n$.
- Names and the :-operator:

$$\mathcal{M}, w \models n_i \text{ iff } w = N_i$$
$$\mathcal{M}, w \models n_1{:}\varphi \text{ iff } \mathcal{M}, N_i \models \varphi$$

The Complete Menu

Our models are structures like
$\mathcal{M} = \langle W, \{R_1, \ldots, R_n\}, \{P_1, \ldots, P_m\}, \{N_1, \ldots, N_k\}\rangle$

- Counting successors:
  $\mathcal{M}, w \models \langle = n\ R\rangle\varphi$ iff $|\{w' \mid wRw'$ and $\mathcal{M}, w' \models \varphi\}| = n$.
- Names and the :-operator:
  $$\mathcal{M}, w \models n_i \text{ iff } w = N_i$$
  $$\mathcal{M}, w \models n_1{:}\varphi \text{ iff } \mathcal{M}, N_i \models \varphi$$
- The $[U]$-operator:
  $$\mathcal{M}, w \models [U]\varphi \text{ iff for all } w', \mathcal{M}, w' \models \varphi$$

The Complete Menu

Our models are structures like
$\mathcal{M} = \langle W, \{R_1, \ldots, R_n\}, \{P_1, \ldots, P_m\}, \{N_1, \ldots, N_k\}\rangle$

- Counting successors:
  $\mathcal{M}, w \models \langle = n\ R\rangle\varphi$ iff $|\{w' \mid wRw'$ and $\mathcal{M}, w' \models \varphi\}| = n$.
- Names and the :-operator:
  $$\mathcal{M}, w \models n_i \text{ iff } w = N_i$$
  $$\mathcal{M}, w \models n_1{:}\varphi \text{ iff } \mathcal{M}, N_i \models \varphi$$
- The $[U]$-operator:
  $$\mathcal{M}, w \models [U]\varphi \text{ iff for all } w', \mathcal{M}, w' \models \varphi$$
- The $[R^*]$-operator:
  $\mathcal{M}, w \models \langle R^*\rangle\varphi$ iff there is w' s.t. wR^*w' and $\mathcal{M}, w' \models \varphi$

The Complete Menu

Our models are structures like
$\mathcal{M} = \langle W, \{R_1, \ldots, R_n\}, \{P_1, \ldots, P_m\}, \{N_1, \ldots, N_k\}\rangle$

- Counting successors:
  $\mathcal{M}, w \models \langle = n\ R\rangle\varphi$ iff $|\{w' \mid wRw'$ and $\mathcal{M}, w' \models \varphi\}| = n$.
- Names and the :-operator:
  $$\mathcal{M}, w \models n_i \text{ iff } w = N_i$$
  $$\mathcal{M}, w \models n_1{:}\varphi \text{ iff } \mathcal{M}, N_i \models \varphi$$
- The $[U]$-operator:
  $$\mathcal{M}, w \models [U]\varphi \text{ iff for all } w', \mathcal{M}, w' \models \varphi$$
- The $[R^*]$-operator:
  $\mathcal{M}, w \models \langle R^*\rangle\varphi$ iff there is w' s.t. wR^*w' and $\mathcal{M}, w' \models \varphi$
- As we can see, the menu is quite varied: GO POLYTHEISM!!!

Back to the Intuitions!

Back to the Intuitions!

The main idea we want to get across is:

There are plenty of options, go and chose what you need!

Back to the Intuitions!

The main idea we want to get across is:

There are plenty of options, go and chose what you need!

Even more, if it is not there, then define it yourself

Back to the Intuitions!

The main idea we want to get across is:

There are plenty of options, go and chose what you need!

Even more, if it is not there, then define it yourself

Remember what Humpty Dumpty said:
"The question is: Which is to be master"

Back to the Intuitions!

The main idea we want to get across is:

There are plenty of options, go and chose what you need!

Even more, if it is not there, then define it yourself

Remember what Humpty Dumpty said:
"The question is: Which is to be master"

- By combining the operators we have been discusing we obtain a wide variety of languages.

Back to the Intuitions!

The main idea we want to get across is:

There are plenty of options, go and chose what you need!

Even more, if it is not there, then define it yourself

Remember what Humpty Dumpty said:
"The question is: Which is to be master"

- By combining the operators we have been discusing we obtain a wide variety of languages.
  - We go from languages of low expressivity (PL) to languages of high expressivity (the selective $[R^*]$).

Back to the Intuitions!

The main idea we want to get across is:

There are plenty of options, go and chose what you need!

Even more, if it is not there, then define it yourself

Remember what Humpty Dumpty said:
"The question is: Which is to be master"

- By combining the operators we have been discusing we obtain a wide variety of languages.
  - We go from languages of low expressivity (PL) to languages of high expressivity (the selective $[R^*]$).
  - We go from languages of 'low' complexity (NP-complete) to languages of hight complexity (EXPTIME-complete).

Back to the Intuitions!

The main idea we want to get across is:

There are plenty of options, go and chose what you need!

Even more, if it is not there, then define it yourself

Remember what Humpty Dumpty said:
"The question is: Which is to be master"

- By combining the operators we have been discusing we obtain a wide variety of languages.
  - We go from languages of low expressivity (PL) to languages of high expressivity (the selective $[R^*]$).
  - We go from languages of 'low' complexity (NP-complete) to languages of hight complexity (EXPTIME-complete).
- By chosing the right expressivity for a given application we will pay the exact price required.

What we Covered Today

What we Covered Today

- We discussed the polytheistic approach in full glory:
  - counting
  - constants
  - universal quantification
  - reflexive and transitive closure

What we Covered Today

- We discussed the polytheistic approach in full glory:
  - counting
  - constants
  - universal quantification
  - reflexive and transitive closure
- By combining all these operators we obtain very diverse logics.

What we Covered Today

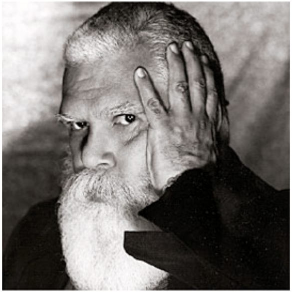
- We discussed the polytheistic approach in full glory:
  - counting
  - constants
  - universal quantification
  - reflexive and transitive closure
- By combining all these operators we obtain very diverse logics.
- But, they all share the same semantics: relational structures!

What we Covered Today

- We discussed the polytheistic approach in full glory:
  - counting
  - constants
  - universal quantification
  - reflexive and transitive closure
- By combining all these operators we obtain very diverse logics.
- But, they all share the same semantics: relational structures!
- They are just different ways of talking about something.

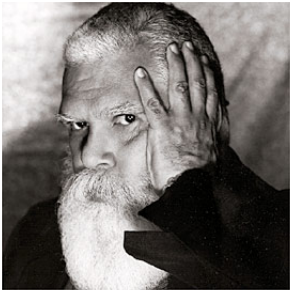
Relevant Bibliography I

[...] No way to say *warm* in French. There was only *hot* and *tepid*. If there's no word for it, how do you think about it? [...] Imagine, in Spanish having to assign a gender to every object: dog, table, tree, can-opener. Imagine, in Hungarian, not being able to assign a gender to anything: *he*, *she*, *it* all the same word.



Relevant Bibliography I

[. . .] No way to say *warm* in French. There was only *hot* and *tepid*. If there's no word for it, how do you think about it? [. . .] Imagine, in Spanish having to assign a gender to every object: dog, table, tree, can-opener. Imagine, in Hungarian, not being able to assign a gender to anything: *he*, *she*, *it* all the same word.



- My French is not good enough to say if it's true. . .
- But it's definitely a great science fiction book!

Delany, Samuel (1966). *Babel-17*. Ace Books.

Relevant Bibliography II

- Many of the languages that we have been discussing are investigated in detail in the area known as Modal Logics.

Relevant Bibliography II

- Many of the languages that we have been discussing are investigated in detail in the area known as Modal Logics.
- The name 'modal' (in many cases as opposed to 'classical') doesn't make much sense.

Relevant Bibliography II

- Many of the languages that we have been discussing are investigated in detail in the area known as Modal Logics.
- The name 'modal' (in many cases as opposed to 'classical') doesn't make much sense.
- Some of these languages have been extensively studied by somebody you know quite well by now.
  Blackburn's Web page: `http://www.loria.fr/~blackbur`

Relevant Bibliography II

- Many of the languages that we have been discussing are investigated in detail in the area known as Modal Logics.
- The name 'modal' (in many cases as opposed to 'classical') doesn't make much sense.
- Some of these languages have been extensively studied by somebody you know quite well by now.
  Blackburn's Web page: `http://www.loria.fr/~blackbur`
- M. de Rijke also pushed the idea of working with modal logics extending the $\langle R \rangle$ language.
  de Rijke's Web page: `http://staff.science.uva.nl/~mdr/`

Relevant Bibliography II

- Many of the languages that we have been discussing are investigated in detail in the area known as Modal Logics.
- The name 'modal' (in many cases as opposed to 'classical') doesn't make much sense.
- Some of these languages have been extensively studied by somebody you know quite well by now.
  Blackburn's Web page: `http://www.loria.fr/~blackbur`
- M. de Rijke also pushed the idea of working with modal logics extending the $\langle R \rangle$ language.

  de Rijke's Web page: `http://staff.science.uva.nl/~mdr/`

Blackburn, Patrick and van Benthem, J (2006). *Chapter 1 of the Handbook of Modal Logics*, Blackburn, P.; Wolter, F.; and van Benthem, J., editors, Elsevier.

de Rijke, Maarten (1993). *Extending Modal Logic*. PhD Thesis. Institute for Logic, Language and Computation, Unviersity of Amsterdam.

The Next Lecture

DIY First Order Logic