A (Modal) Logic Toolkit

Carlos Areces

carlos.areces@unc.edu.ar

UNC, CONICET, & GTIIT

CLAWS 2025 - Guangdong - CHINA

What is Logic, and why should I care?

- Probably all of you have heard about 'Logic' before.
- But what is logic for you? Perhaps it's the science that studies strange symbols like

$$(p \wedge q) \rightarrow (p \vee q)$$

$$\forall x(\text{Human}(x) \rightarrow \text{Mortal}(x))$$

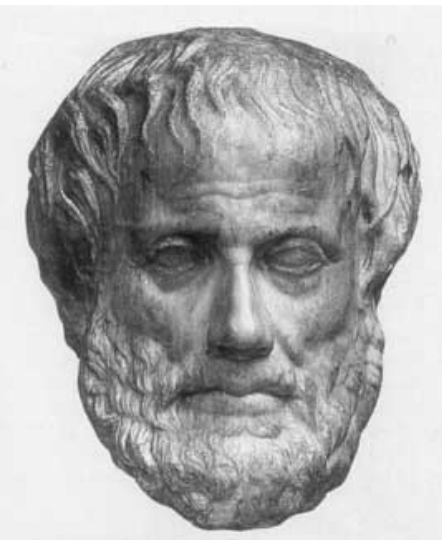
that are (allegedly) important in computer science, philosophy, etc., for some (mysterious) reasons.

- And perhaps you've encountered what logicians called 'theorems', expressions like:

$$p \vee \neg p \quad \text{or} \quad p \rightarrow p.$$

Back to Aristotle

Or perhaps you have met logic in a philosophical setting? You're aware of the work of Aristotle (384 BC - 322 BC), and in particular his discussion of syllogisms.



All vampires are demons.
Angel is a vampire.
Therefore Angel is a demon.

Obscure reference to cult TV series, check!

Tomorrow it will rain or it won't...

Either way, logic may not have struck you as particularly exciting or relevant to your work.

- Sentences like "John loves Mary, or not" or "It will rain or it won't, tomorrow" sound a bit silly. They don't seem to be very informative.

Tomorrow it will rain or it won't...

Either way, logic may not have struck you as particularly exciting or relevant to your work.

- Sentences like "John loves Mary, or not" or "It will rain or it won't, tomorrow" sound a bit silly. They don't seem to be very informative.
- Nor do simple syllogisms seem to have much to with with the kind of reasoning we do everyday.

Tomorrow it will rain or it won't...

Either way, logic may not have struck you as particularly exciting or relevant to your work.

- Sentences like "John loves Mary, or not" or "It will rain or it won't, tomorrow" sound a bit silly. They don't seem to be very informative.
- Nor do simple syllogisms seem to have much to with with the kind of reasoning we do everyday.
- And they certainly seem far removed from the type of arguments found in computer science and mathematics. And the mathematicians notion of 'theorem' seems very different (and much richer) than the logicians notion.

Tomorrow it will rain or it won't...

Either way, logic may not have struck you as particularly exciting or relevant to your work.

- Sentences like "John loves Mary, or not" or "It will rain or it won't, tomorrow" sound a bit silly. They don't seem to be very informative.
- Nor do simple syllogisms seem to have much to with with the kind of reasoning we do everyday.
- And they certainly seem far removed from the type of arguments found in computer science and mathematics. And the mathematicians notion of 'theorem' seems very different (and much richer) than the logicians notion.
- I will try to set this picture straight (at least a bit) in this tutorial.

Logics as Languages

- I want you to think of logics as languages.
- In particular I want you to think of logics as ways of talking about relational structures (our *models*).
- That is, there are two key components in the way we will approach logic
  - The logic: fairly simple, precisely defined, formal languages. (This is where the funny symbols like $\wedge$ and $\exists$ live).
  - The model or relational structure: A simple 'world' (or 'database') that the logic talks about.

Semantic perspective

That is, our perspective on logic is fundamentally semantic. It is due to Alfred Tarski (1902–1983).

The semantic perspective is also known as the model-theoretic perspective, or even the Tarskian perspective.

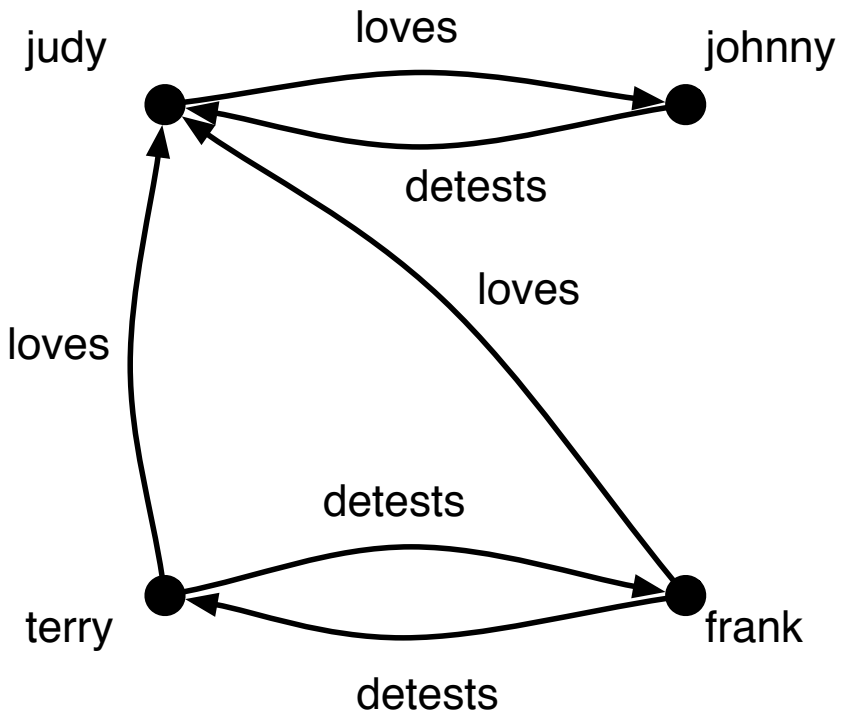
Cute picture of logician, check!

Relational structures (informal)

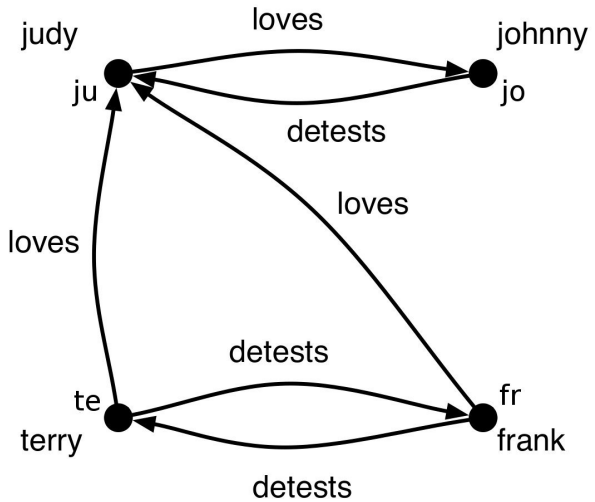
A relational structure (or model) consists of the following

- A non-empty set (often called D, for domain) of the model; think of these as the objects of interest.
- A collection of relations R on the objects in D; think of these as the relations of interest (we shall only work with binary relations (that is, two place relations like "loves", "$\leq$", or "to-the-right-of" in this course) to keep the notation simple.
- A collection of properties on the objects in D; think of these as the properties of interests (perhaps "is red", "is activated", or "is an even number").
- A collection of designated individuals, that is, elements of D that we find really special (maybe "Buffy", "0", or "1").

Our first relational structure

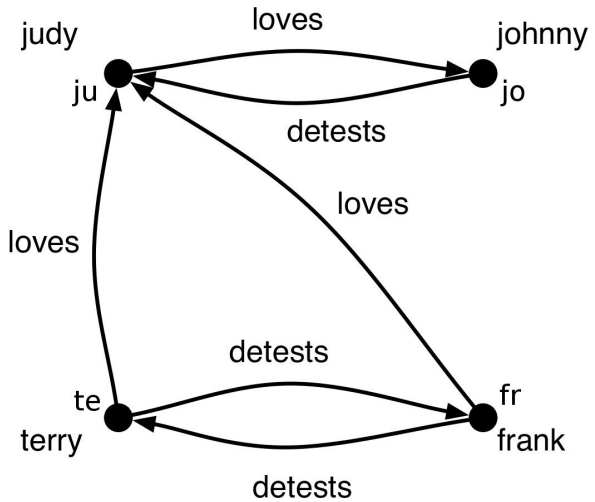


Another look at our first relational structure



$$\langle D, \{loves, detest\}, \emptyset, C \rangle$$

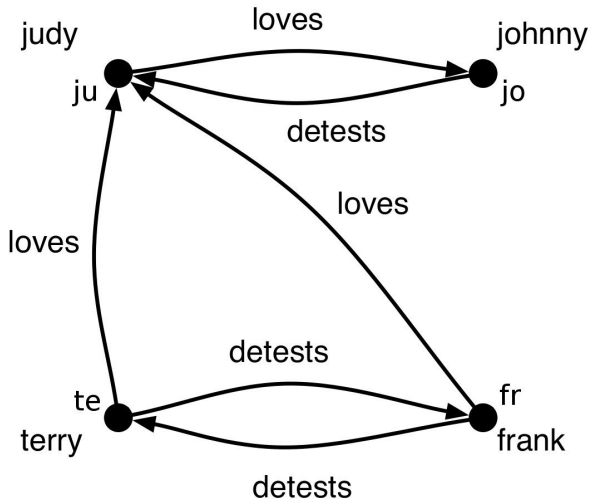
Another look at our first relational structure



$$\langle D, \{loves, detest\}, \emptyset, C \rangle$$

$$D = \{ju, jo, te, fr\}$$

Another look at our first relational structure

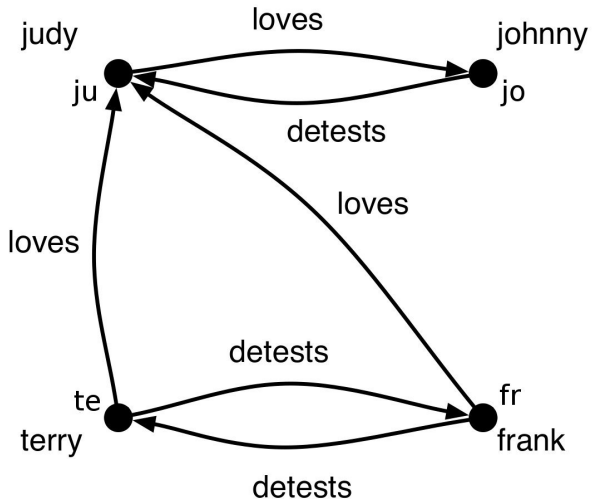


$$\langle D, \{loves, detest\}, \emptyset, C \rangle$$

$D = \{ju, jo, te, fr\}$

$loves = \{(ju, jo), (te, ju), (fr, ju), \}$

Another look at our first relational structure



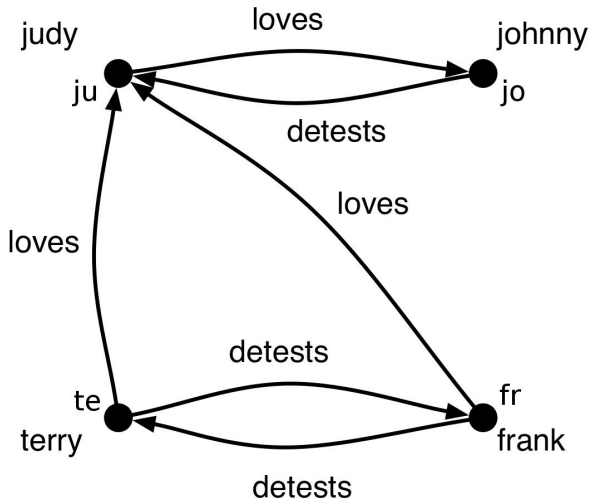
$$\langle D, \{loves, detest\}, \emptyset, C \rangle$$

$D = \{ju, jo, te, fr\}$

$loves = \{(ju, jo), (te, ju), (fr, ju), \}$

$detest = \{(jo, ju), (te, fr), (fr, te), \}$

Another look at our first relational structure



$$\langle D, \{loves, detest\}, \emptyset, C \rangle$$

$D = \{ju, jo, te, fr\}$

$loves = \{(ju, jo), (te, ju), (fr, ju), \}$

$detest = \{(jo, ju), (te, fr), (fr, te), \}$

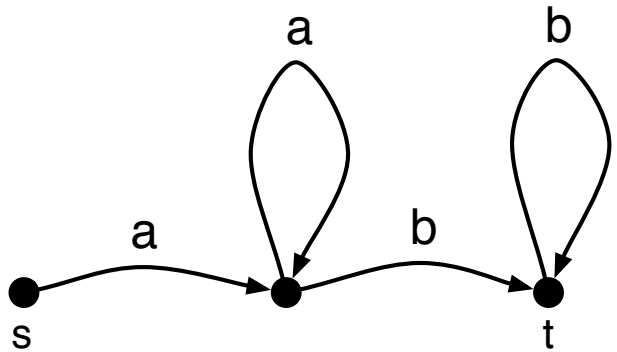
$C = \{judy \mapsto ju, johnny \mapsto jo, terry \mapsto te, frank \mapsto fr\}$

What can be thought of as a relational structure. . . ?

That's the wrong question — the real question is, what can't be thought of as a relational structure?

In fact, it is very hard to think of anything (barring some rather extreme mathematical examples) that can't be viewed as a relational structure.

Example



This shows a finite state automaton for the formal language $a^n b^m$ $(n, m > 0)$, that is, for the set of all strings consisting of a non-empty block of as followed by a non-empty block of bs.

A general and important modelling tool

- All common mathematical structures can be though of as relational structures. For a start, functions are simply special kinds of relations.
- Moreover, groups, rings, field, vector spaces, . . . can all be viewed as relational structures.
- Thus Tarski's idea had substantial mathematical impact, and this was decisive in establishing model-theory as an academic discipline.

The three big themes

In this tutorial I will use the model-theoretic perspective to briefly introduce the following three themes:

- Inference: roughly speaking, what methods are there for gaining new information by working with this logic?
- Expressivity: roughly speaking, what can I describe (and what can't I describe) using this logic?
- Computation: too many to list — can computers help with this logic? If so how, and how much?

Inference tasks

The semantic perspective provides a good way to think about inference tasks. I can mention the following three:

- Given a certain formula φ, and a model $\mathcal{M}$, does φ correctly describe (some aspect of) the model? More simply: is the formulas true (or satisfied) in the model? This task is called the model checking task.
- Given a certain formula φ, does there exist a model where φ is true? This task is called the satisfiability checking task (or the model building task).
- Given a certain formula φ, is it true (or satisfied) in all models? This task is called the validity checking task.

The Model Checking task

- This is the simplest of the three tasks.
- It has also proved to be one of the most useful.
- A classic application is hardware verification. The model $\mathcal{M}$ is a mathematical picture of (say) a chip. The logical description φ describes some desirable feature of of the chip. If the $\mathcal{M}$ makes φ true, then the chip will have that property.
- Incidentally, this example already suggests the need for "designing logics for their application". After all, there is not reason to think that an off-the-shelf logic will provide exactly what is needed to talk usefully about chips and their properties.

Satisfiability Checking (and Model Building)

- A nice way to think of this problem is in terms of constraints. We have some description. Is there anything that matches this description? That is, does a model making this description actually exist, and can we build it?

Satisfiability Checking (and Model Building)

- A nice way to think of this problem is in terms of constraints. We have some description. Is there anything that matches this description? That is, does a model making this description actually exist, and can we build it?
- Very useful. The description might be almost anything: for example, a description of your dream partner. Then, use model building to find Love!

Validity Checking

- A great deal of attention has been devoted to this task — essentially, when people talk about "writing a theorem prover," they are talking about creating a computational tool for solving this task.
- The real question is: why? After all, we've already mentions that $p \vee \neg p$ and $p \rightarrow p$ are not going to set too many pulses racing...

Logic as a tool for working with theories

- Let's turn to mathematics. The intuitive idea is that we write down a set Σ of all our axioms. These are the properties that we assume are fundamental and indisputable; what we take for granted. Σ is our theory.
- For example Peano axioms are a theory for the natural numbers.
- Checking if the Goldbach Conjecture is true in the natural numbers boils down to verifying that

$$(\bigwedge \text{PEANO}) \rightarrow \text{GOLDBACH}$$

is a 'trivial" formula, that is, a validity.

GOLDBACH = every even integer > 2 is the sum of two prime numbers.
Cool reference to important open mathematical problem, check!

Expressivity

- The theme of expressivity is fundamental — specially to a model-theoretically inclined logician — though difficult to explain.
- The fundamental point is this. Once we have said which relational structures we are interested in, there are many logics suitable for talking about them. Each offers a different (often a fascinatingly different) perspective on the same "world". Which one we choose? Which is best?

Expressivity

- The theme of expressivity is fundamental — specially to a model-theoretically inclined logician — though difficult to explain.
- The fundamental point is this. Once we have said which relational structures we are interested in, there are many logics suitable for talking about them. Each offers a different (often a fascinatingly different) perspective on the same "world". Which one we choose? Which is best?
- As with most fundamental questions in life, the answer is "It depends". And the important question is to look for "it depends on **what**"?

Computer Science and Logic

Here I just refer back to my opening talk.

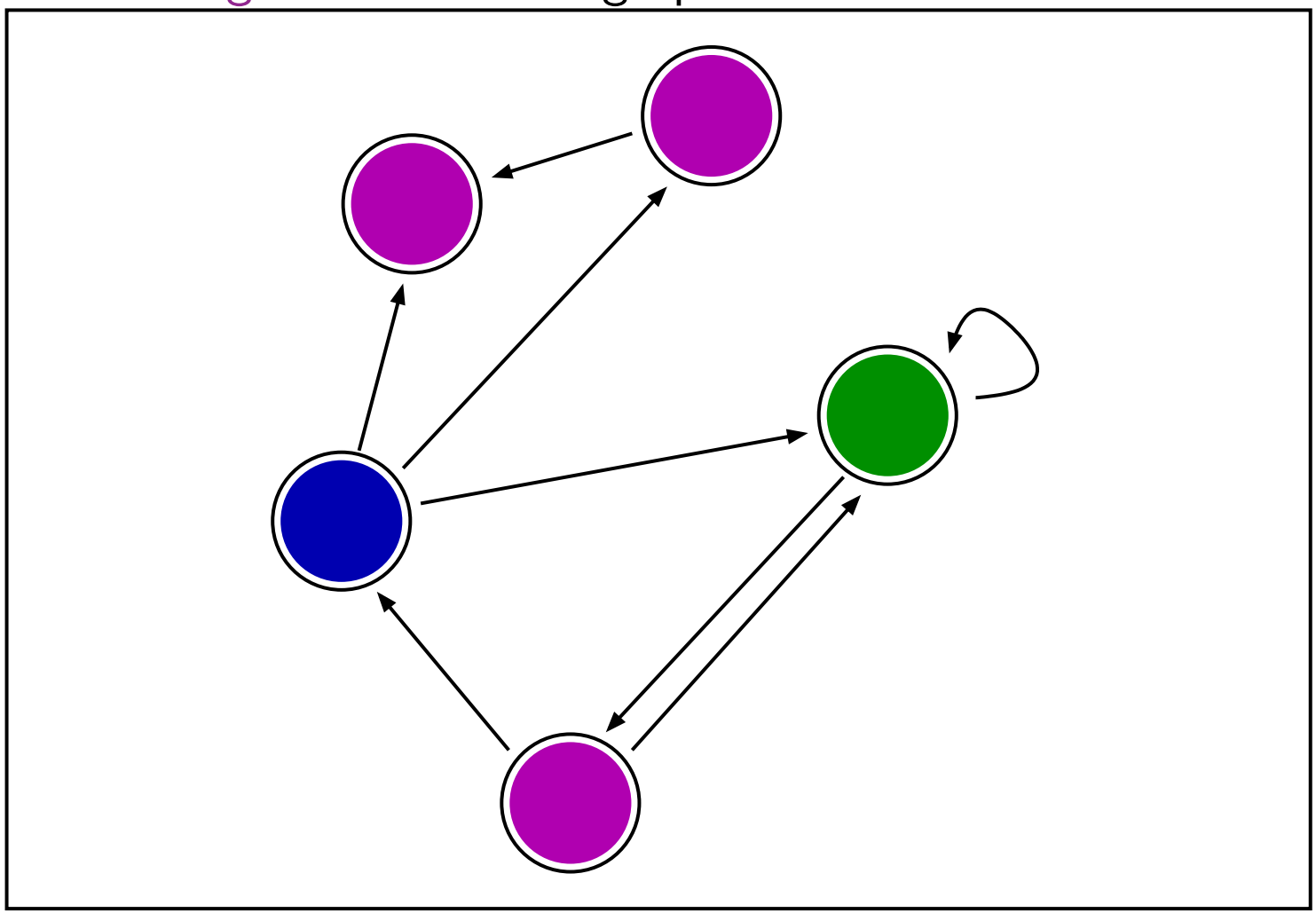
- First, ideas from theoretical computer science (such as computational complexity) are fundamental tools for analyzing logics.
- Second, more and more computer science is setting the agenda in logic.
- Third, at a practical level we simply need computers when working with logic.

Simple Structures / Simple Languages

Think of standing in this colored graph:

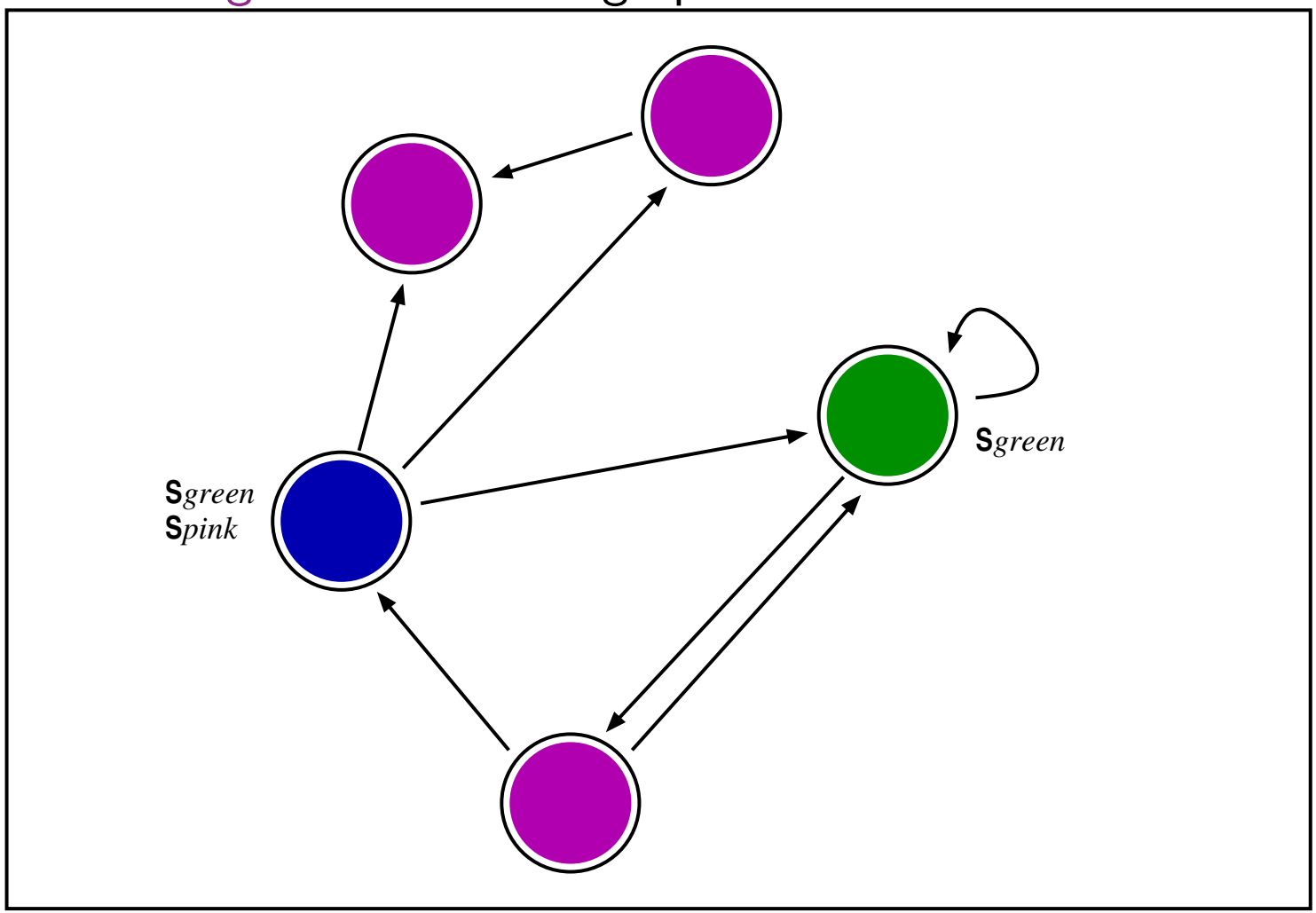
Simple Structures / Simple Languages

Think of standing in this colored graph:



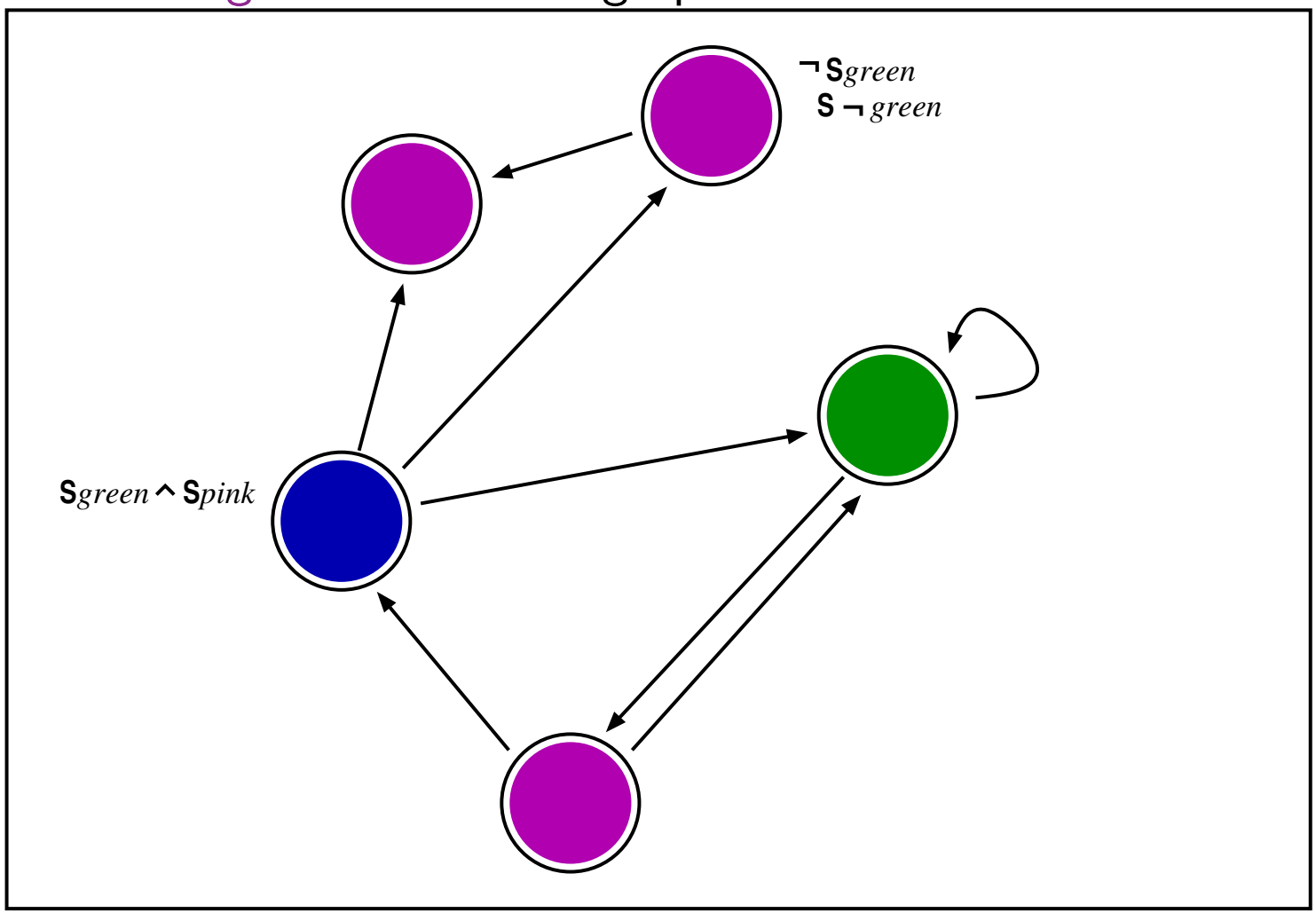
Simple Structures / Simple Languages

Think of standing in this colored graph:



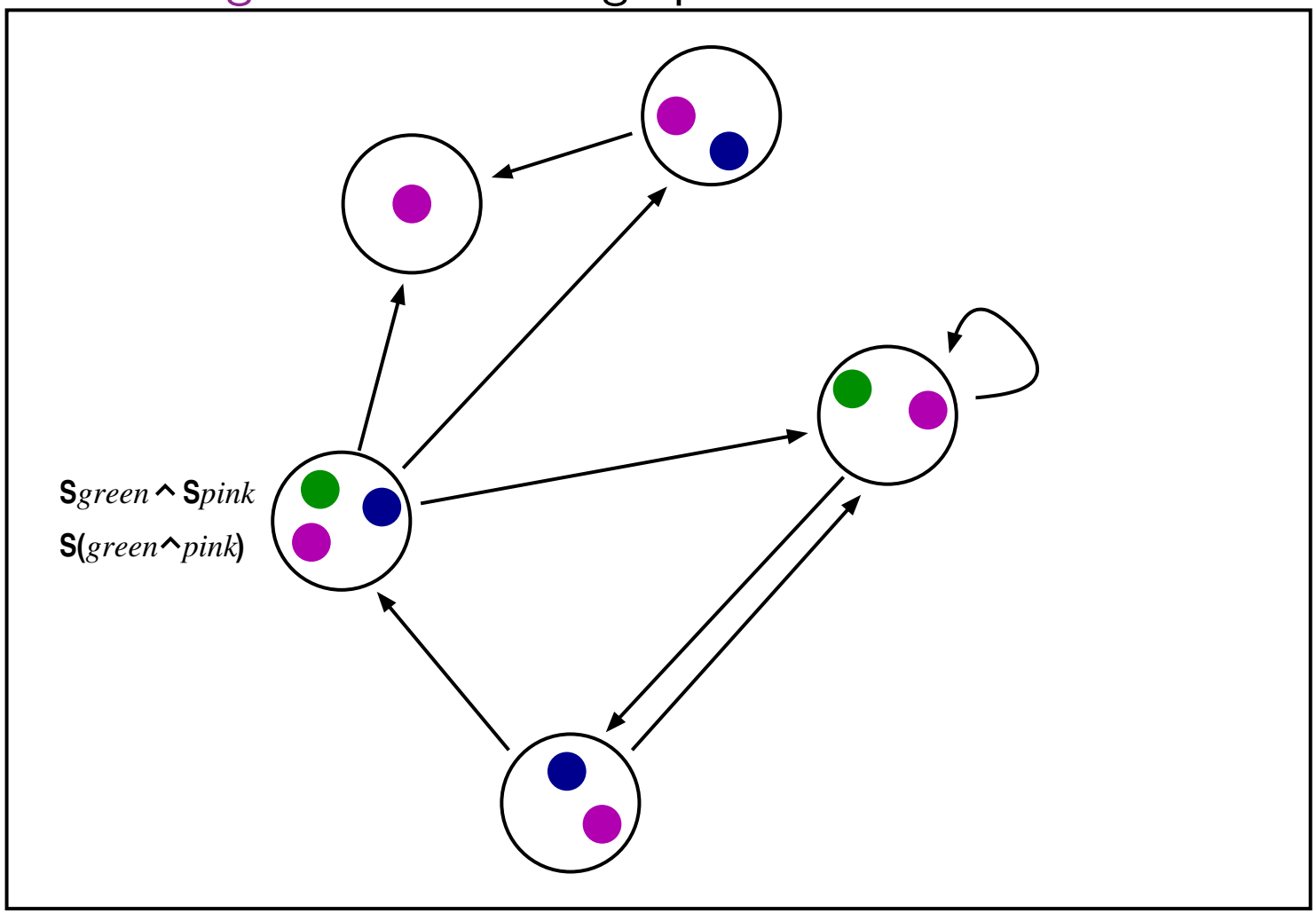
Simple Structures / Simple Languages

Think of standing in this colored graph:



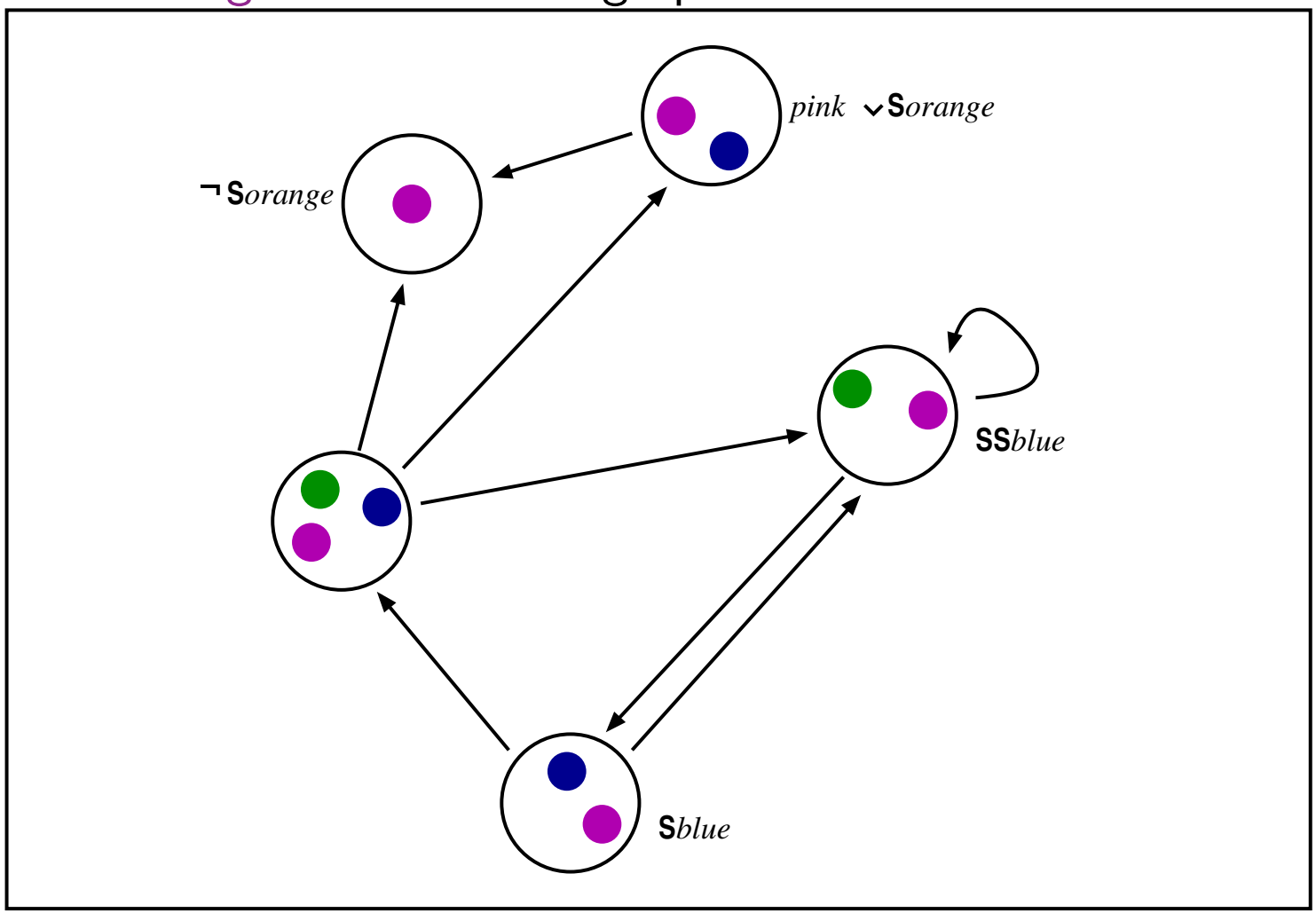
Simple Structures / Simple Languages

Think of standing in this colored graph:



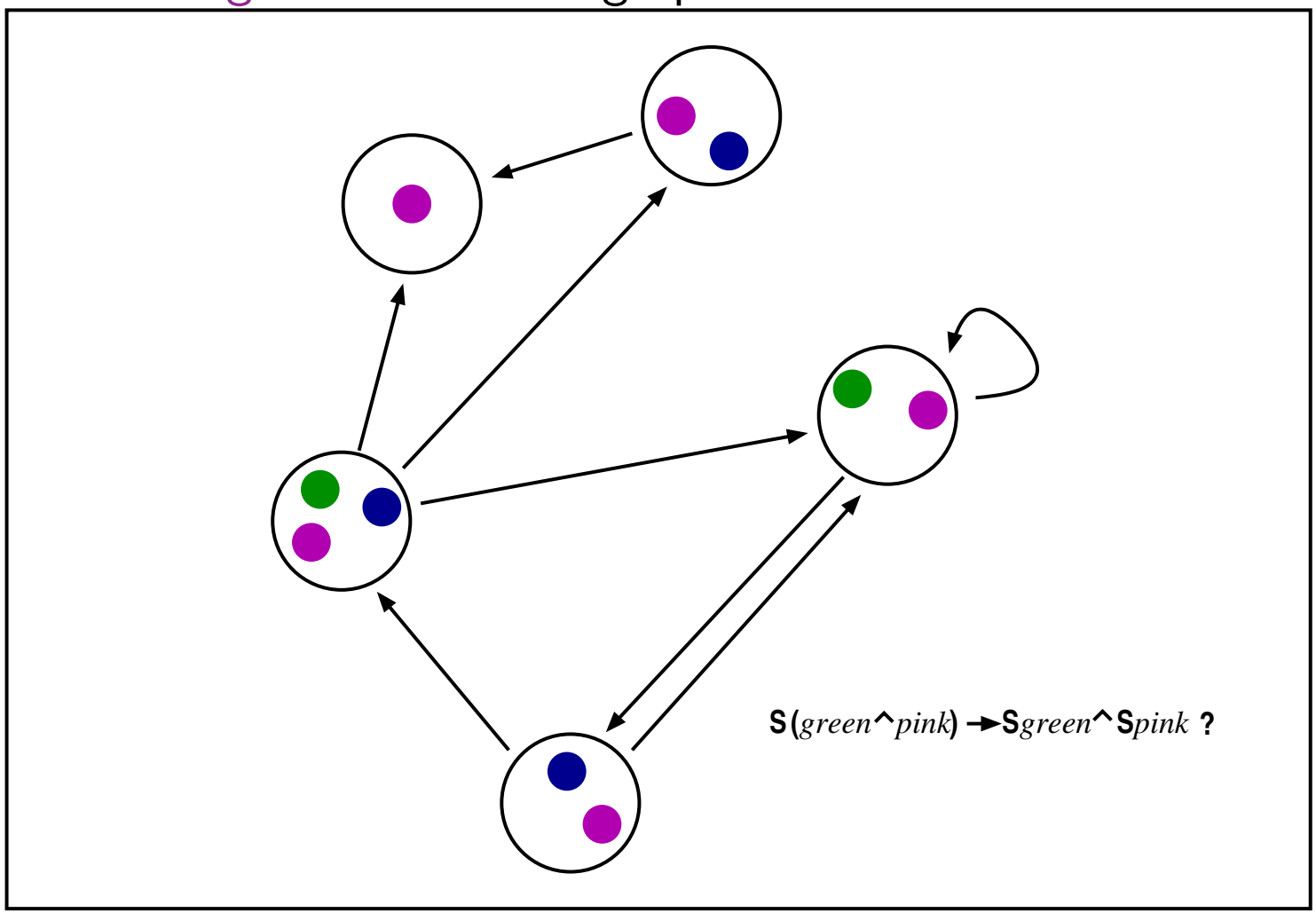
Simple Structures / Simple Languages

Think of standing in this colored graph:



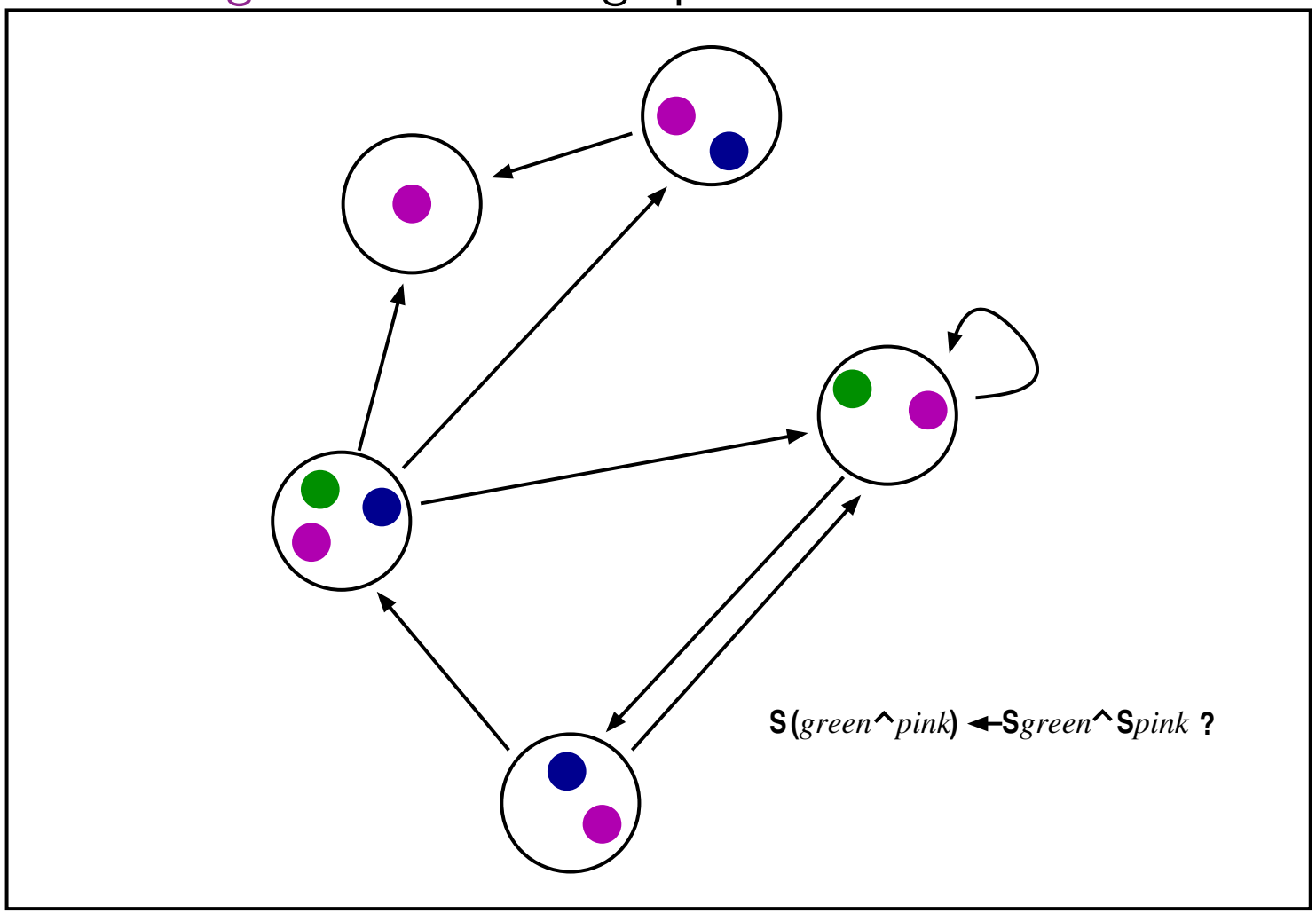
Simple Structures / Simple Languages

Think of standing in this colored graph:



Simple Structures / Simple Languages

Think of standing in this colored graph:



Basic Modal Language: Syntax

We build the modal language on top of PL. It is a very simple extension. First we decide how many relations R we want to work with, and then we add the following two symbols for each R:

$$\langle R \rangle \qquad [R]$$

If we are only going to work with a single relation, we usually write these as:

$$\Diamond \qquad \Box$$

We then extend the definition of formula by saying that if φ is a formula, then so are $\langle R \rangle \varphi$ and $[R]\varphi$.

Basic Modal Language: Semantics

Suppose we are given a relational structure

$$\langle W, \{R_1, \ldots, R_n\}, \{P_1, \ldots, P_m\}\rangle$$

where we have one relation R for each diamond $\langle R\rangle$, and one property P for each proposition symbol p. Then we define:

$$\mathcal{M}, w \models p_i \qquad \text{iff} \qquad w \in P_i,$$

Basic Modal Language: Semantics

Suppose we are given a relational structure

$$\langle W, \{R_1, \ldots, R_n\}, \{P_1, \ldots, P_m\}\rangle$$

where we have one relation R for each diamond $\langle R\rangle$, and one property P for each proposition symbol p. Then we define:

$$\begin{array}{rcl}
\mathcal{M}, w \models p_i & \text{iff} & w \in P_i, \\
\mathcal{M}, w \models \neg\varphi & \text{iff} & \mathcal{M}, w \not\models \varphi, \\
\mathcal{M}, w \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi,
\end{array}$$

Basic Modal Language: Semantics

Suppose we are given a relational structure

$$\langle W, \{R_1, \ldots, R_n\}, \{P_1, \ldots, P_m\}\rangle$$

where we have one relation R for each diamond $\langle R\rangle$, and one property P for each proposition symbol p. Then we define:

$$\begin{array}{rll}
\mathcal{M}, w \models p_i & \text{iff} & w \in P_i, \\
\mathcal{M}, w \models \neg\varphi & \text{iff} & \mathcal{M}, w \not\models \varphi, \\
\mathcal{M}, w \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi, \\
\mathcal{M}, w \models \langle R_j\rangle\varphi & \text{iff} & \text{for some } v \in W \text{ such that } (w, v) \in R_j \\
 & & \text{we have } \mathcal{M}, v \models \varphi,
\end{array}$$

Basic Modal Language: Semantics

Suppose we are given a relational structure

$$\langle W, \{R_1, \ldots, R_n\}, \{P_1, \ldots, P_m\}\rangle$$

where we have one relation R for each diamond $\langle R\rangle$, and one property P for each proposition symbol p. Then we define:

$$\begin{array}{rll}
\mathcal{M}, w \models p_i & \text{iff} & w \in P_i, \\
\mathcal{M}, w \models \neg\varphi & \text{iff} & \mathcal{M}, w \not\models \varphi, \\
\mathcal{M}, w \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi, \\
\mathcal{M}, w \models \langle R_j\rangle\varphi & \text{iff} & \text{for some } v \in W \text{ such that } (w, v) \in R_j \\
 & & \text{we have } \mathcal{M}, v \models \varphi, \\
\mathcal{M}, w \models [R_j]\varphi & \text{iff} & \text{for all } v \in W \text{ such that } (w, v) \in R_j \\
 & & \text{we have } \mathcal{M}, v \models \varphi.
\end{array}$$

We don't need both boxes and diamonds

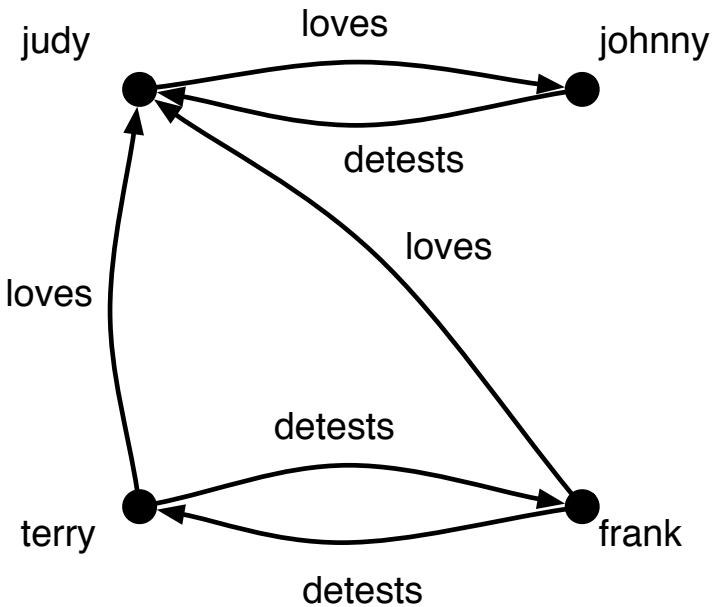
$\Box\varphi$ is $\neg\Diamond\neg\varphi$.

$\Diamond\varphi$ is $\neg\Box\neg\varphi$.

(So, like WallE, we can choose the box and throw the diamond.)

Reference to beloved movie, check!

Love and hate . . .



$$\langle \text{LOVES} \rangle \top \wedge \langle \text{DETESTS} \rangle \langle \text{LOVES} \rangle \top$$

Note that this is true when evaluated at Terry

Fundamental semantic concepts

- A formula φ is satisfiable in a model $\mathcal{M}$ if there is some point in $\mathcal{M}$ at which φ is satisfied, and φ is satisfiable if there is some point in some model at which it is satisfied.

Fundamental semantic concepts

- A formula φ is satisfiable in a model $\mathcal{M}$ if there is some point in $\mathcal{M}$ at which φ is satisfied, and φ is satisfiable if there is some point in some model at which it is satisfied.
- A formula φ is valid if it is globally satisfied in all models, and if this is the case we write $\models \varphi$.

Fundamental semantic concepts

- A formula φ is satisfiable in a model $\mathcal{M}$ if there is some point in $\mathcal{M}$ at which φ is satisfied, and φ is satisfiable if there is some point in some model at which it is satisfied.
- A formula φ is valid if it is globally satisfied in all models, and if this is the case we write $\models \varphi$.
  (A formula φ is globally satisfied (or globally true) in a model $\mathcal{M}$ if it is satisfied at all points in $\mathcal{M}$, and if this is the case we write $\mathcal{M} \models \varphi$.)

Three examples of validities

The following three formulas are validities:

- $\langle R \rangle (p \wedge q) \rightarrow \langle R \rangle p \wedge \langle R \rangle q$
- $\langle R \rangle \top \wedge \neg \langle R \rangle \neg p \rightarrow \langle R \rangle p$
- $[R](p \rightarrow q) \rightarrow ([R]p \rightarrow [R]q)$ (Axiom K)

Can you see why? Note: you've seen the first one already ...

Axiomatizing the BML

1. Take Valentin's work. I.e., axioms A1 to A7 plus the (mp) rule for PL
2. For the modal part
   - 2.1 K $[R](p \to q) \to ([R]p \to [R]q)$
   - 2.2 (nec) rule: From φ infer $[R]\varphi$

Inference and computation in the basic modal language

- It should be clear that defining an inference systems (such as tableaux systems) for the modal language is going to be trickier (and more interesting!) than for PL.
- It should also be clear that there are computational issues to be settled — there is no obvious way to prove that satisfiability in the modal language is computable by "listing all models" as is done with truth tables in PL.
- That is, we have traded in some tractability for expressivity.

Model checking I

Use a bottom-up labeling algorithm. To model check a formula φ:

- Label every point in the model with all the subformulas of φ that are true at that point.
- We start with the proposition symbols: the valuation tells us where these are true, so we label all the appropriate points.
- We then label with more complex formulas. The booleans are handled in the obvious way: for example, we label w with $\psi \wedge \theta$ if w is labeled with both ψ and θ.
- As for the modalities, we label w with $\Diamond\varphi$ if one of its R-successors is labeled with φ, and we label it with $\Box\varphi$ if all of its R-successors are labeled with φ.

Model checking II

- The beauty of this algorithm is that we never need to duplicate work: once a point is labeled as making φ true, that's it.
- This makes the algorithm run in time polynomial in the size of the input formula and model: the algorithm takes time of the order of

$$con(\varphi) \times nodes(\mathcal{M}) \times nodes(\mathcal{M}),$$

  where $con(\varphi)$ is the number of connectives in φ, and $nodes(\mathcal{M})$ is the number of nodes in $\mathcal{M}$.
- To see this, note that $con(\varphi)$ tells us how many rounds of labeling we need to perform, one of the $nodes(\mathcal{M})$ factors is simply the upper bound on the nodes that need to be labeled, while the other is the upper bound on the number of successor nodes that need to be checked.

Model checking is important

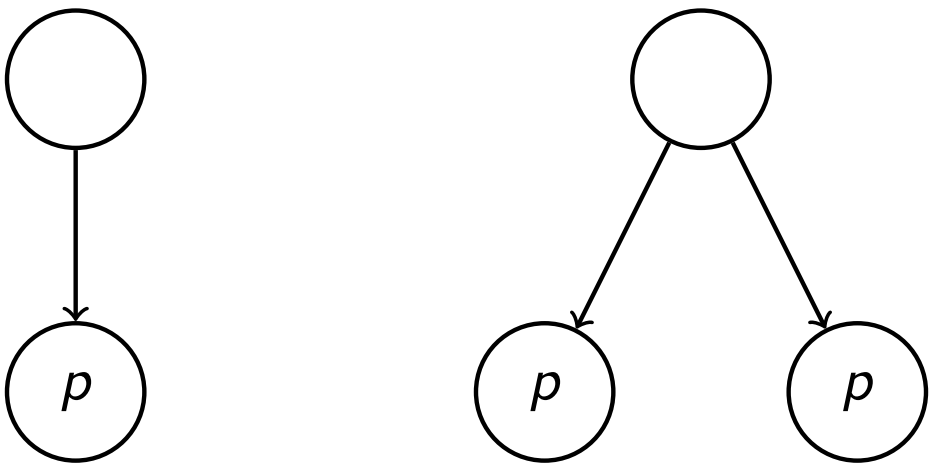
- Although this looks simple, this kind of model checking is important.
- It is possible to enrich the modal language in various ways to make it excellent for talking about graph structures representing hardware chips and much else besides.
- The above algorithm can (and have been) extended to such languages and applied to industrial hardware and software design problems.
- Model checking is the simplest form of inference — but it is important and useful.

Expressivity and bisimulation

- What can we say in the basic modal language? And what can't we say? That is, how expressive is our modal language?
- We will ask ourselves the following question: how good are modal languages at distinguishing between models?
- Let's look at an example...

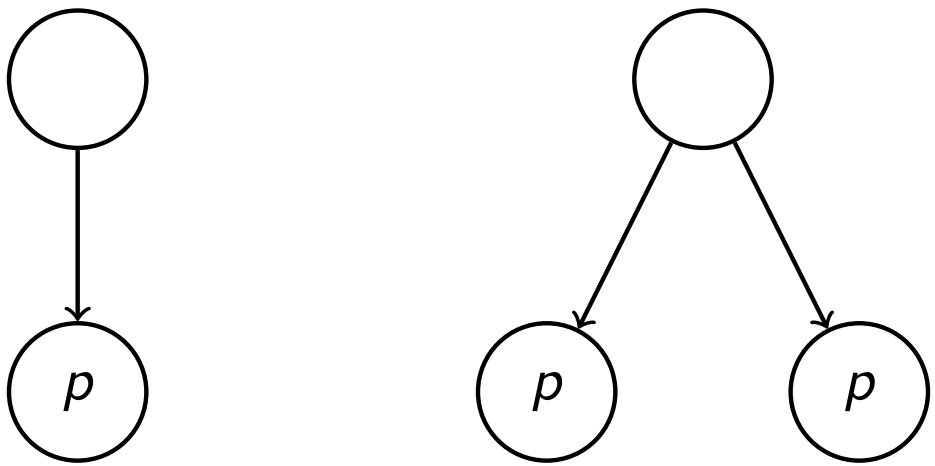
Are these model distinguishable in BML?

Are these two models distinguishable in our modal language? To make the question more precise: is there an modal formula that is true at the root node of one model, and not at the other?



Are these model distinguishable in BML?

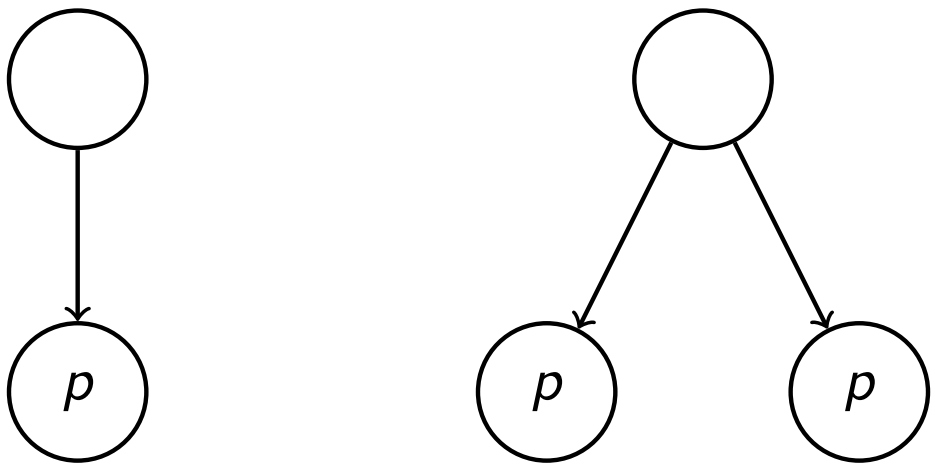
Are these two models distinguishable in our modal language? To make the question more precise: is there an modal formula that is true at the root node of one model, and not at the other?



No. There is no modal formula that distinguishes them. The modal language, it seems, cannot count!

Are these model distinguishable in BML?

Are these two models distinguishable in our modal language? To make the question more precise: is there an modal formula that is true at the root node of one model, and not at the other?



No. There is no modal formula that distinguishes them. The modal language, it seems, cannot count!

Now the key question: why exactly can't the modal language distinguish the two models?

Bisimulations (informal)

Two models are bisimilar if their points can be related in in such a way that:

- Related points make the same propositional symbols true.
- If you make a transition in one model, you can make "matching" transition in the other. Here "matching" means that the points you reach by making the transitions are related.

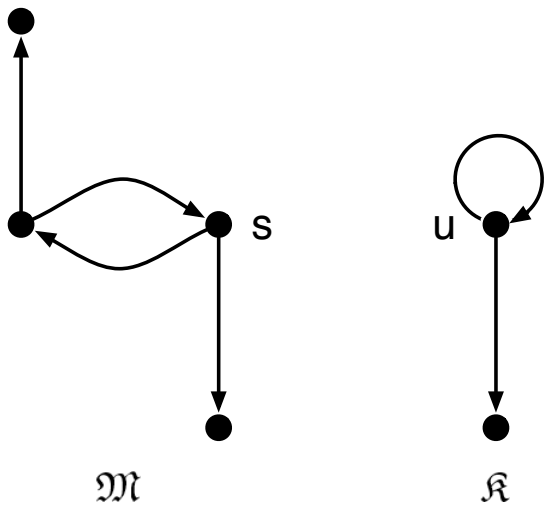
Bisimulations (formal definition)

Here is th formal definition of bisimulation (for models with one relation R). A bisimulation between models $\mathcal{M} = (W, R, V)$ and $\mathcal{M}' = (W', R', V')$ is a non-empty binary relation E between their domains (that is, $E \subseteq W \times W'$) such that whenever $(w, w') \in E$ we have that:

> **Atomic harmony:** w and w' satisfy the same proposition symbols,
>
> **Zig:** if $(w, v) \in R$, then there exists a point v' (in $\mathcal{M}'$) such that $(w', v') \in R'$ and $(v, v') \in E$, and
>
> **Zag:** if $(w', v') \in R'$, then there exists a point v (in $\mathcal{M}$) such that $(w, v) \in R$ and $(v, v') \in E$.

Bisimilar or not?

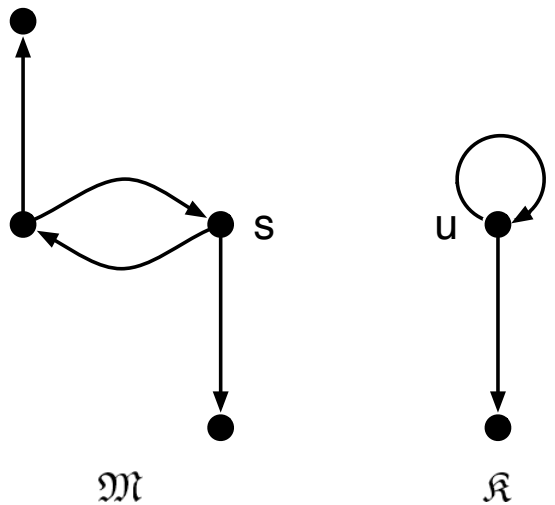
Are points s and u in these two models bisimilar or not (assume all propositional symbols are false in all points)?



If they are bisimilar, what is the bisimulation? If they are not bisimilar, what is a formula that distinguishes them?

Bisimilar or not?

Are points s and u in these two models bisimilar or not (assume all propositional symbols are false in all points)?

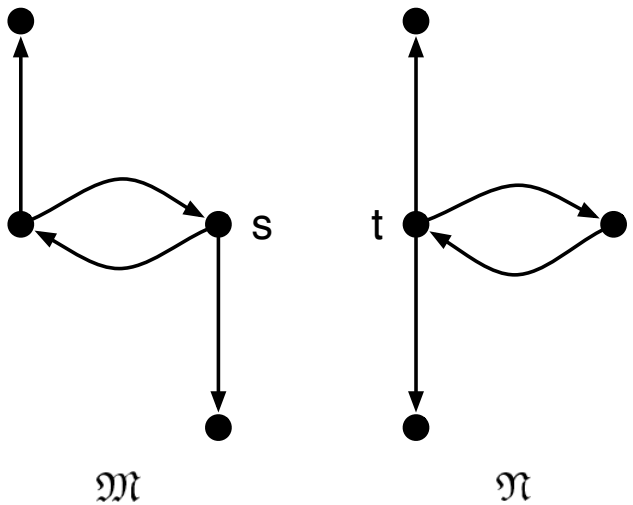


If they are bisimilar, what is the bisimulation? If they are not bisimilar, what is a formula that distinguishes them?

Yes, they are bisimilar; to see this, bend the upward-pointing arrow on the left of the left-hand model downwards.

Bisimilar or not?

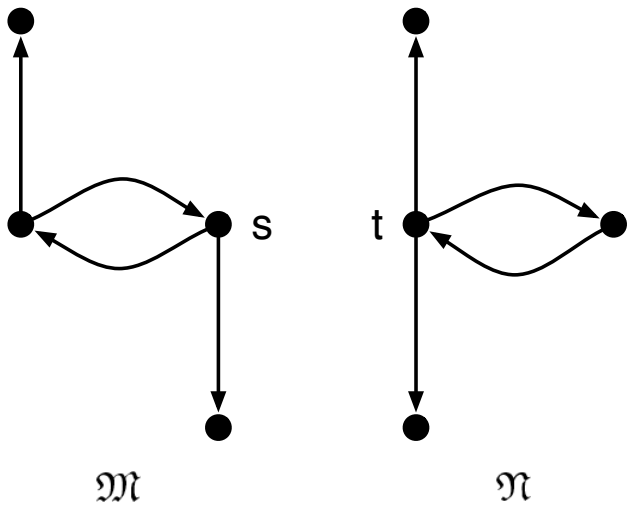
Are points s and t in these two models bisimilar or not (assume all propositional symbols are false in all points)?



If they are bisimilar, what is the bisimulation? If they are not bisimilar, what is a formula that distinguishes them?

Bisimilar or not?

Are points s and t in these two models bisimilar or not (assume all propositional symbols are false in all points)?



If they are bisimilar, what is the bisimulation? If they are not bisimilar, what is a formula that distinguishes them?

$\Box(\Box\bot \vee \Diamond\Box\bot)$ is a formula that distinguishes these models: it is true in $\mathcal{M}$ at s, but false in $\mathcal{N}$ at t.

Useful bisimulations

Bisimulations tell us when two relational structures are equivalent for the modal language. And there is an interesting connection between them and Automata Theory:

- Hopcroft' minimization algorithms returns, given a deterministic finite automata (DFA), an equivalent DFA of minimal size.

Useful bisimulations

Bisimulations tell us when two relational structures are equivalent for the modal language. And there is an interesting connection between them and Automata Theory:

- Hopcroft' minimization algorithms returns, given a deterministic finite automata (DFA), an equivalent DFA of minimal size.
- Automata minimization is important, because the complexity of many computational tasks over automata (i.e., determining if the language of the automata is empty) are parametric on the size of the automata.

Useful bisimulations

Bisimulations tell us when two relational structures are equivalent for the modal language. And there is an interesting connection between them and Automata Theory:

- Hopcroft' minimization algorithms returns, given a deterministic finite automata (DFA), an equivalent DFA of minimal size.
- Automata minimization is important, because the complexity of many computational tasks over automata (i.e., determining if the language of the automata is empty) are parametric on the size of the automata.
- If we see the automata as a relational structure, then computing minimization is equivalent to collapse states in the model in terms of its largest autobisimulation.

Useful bisimulations

Bisimulations tell us when two relational structures are equivalent for the modal language. And there is an interesting connection between them and Automata Theory:

- Hopcroft' minimization algorithms returns, given a deterministic finite automata (DFA), an equivalent DFA of minimal size.
- Automata minimization is important, because the complexity of many computational tasks over automata (i.e., determining if the language of the automata is empty) are parametric on the size of the automata.
- If we see the automata as a relational structure, then computing minimization is equivalent to collapse states in the model in terms of its largest autobisimulation.

Connection between modal logic and important algorithm in CS, check!

Summary of the tutorial

In this tutorial we covered the following:

1. We presented a model-theoretic perspective of Logic, by presenting them as languages to describe relational structures.

Summary of the tutorial

In this tutorial we covered the following:

1. We presented a model-theoretic perspective of Logic, by presenting them as languages to describe relational structures.
2. We introduce fundamental inference tasks in Logic: model checking, satisfiability checking, and validity checking.

Summary of the tutorial

In this tutorial we covered the following:

1. We presented a model-theoretic perspective of Logic, by presenting them as languages to describe relational structures.
2. We introduce fundamental inference tasks in Logic: model checking, satisfiability checking, and validity checking.
3. We stressed the role of Logic in CS and vice-versa.

Summary of the tutorial

In this tutorial we covered the following:

1. We presented a model-theoretic perspective of Logic, by presenting them as languages to describe relational structures.
2. We introduce fundamental inference tasks in Logic: model checking, satisfiability checking, and validity checking.
3. We stressed the role of Logic in CS and vice-versa.
4. We introduced Modal Logics as a simple language to describe relational structures.

Summary of the tutorial

In this tutorial we covered the following:

1. We presented a model-theoretic perspective of Logic, by presenting them as languages to describe relational structures.
2. We introduce fundamental inference tasks in Logic: model checking, satisfiability checking, and validity checking.
3. We stressed the role of Logic in CS and vice-versa.
4. We introduced Modal Logics as a simple language to describe relational structures.
5. We presented the syntax and semantics of BML.

Summary of the tutorial

In this tutorial we covered the following:

1. We presented a model-theoretic perspective of Logic, by presenting them as languages to describe relational structures.
2. We introduce fundamental inference tasks in Logic: model checking, satisfiability checking, and validity checking.
3. We stressed the role of Logic in CS and vice-versa.
4. We introduced Modal Logics as a simple language to describe relational structures.
5. We presented the syntax and semantics of BML.
6. We discussed model checking and axiomatization for BML.

Summary of the tutorial

In this tutorial we covered the following:

1. We presented a model-theoretic perspective of Logic, by presenting them as languages to describe relational structures.
2. We introduce fundamental inference tasks in Logic: model checking, satisfiability checking, and validity checking.
3. We stressed the role of Logic in CS and vice-versa.
4. We introduced Modal Logics as a simple language to describe relational structures.
5. We presented the syntax and semantics of BML.
6. We discussed model checking and axiomatization for BML.
7. We introduced bisimulation as the tool to measure when two relations are equivalent for BML.