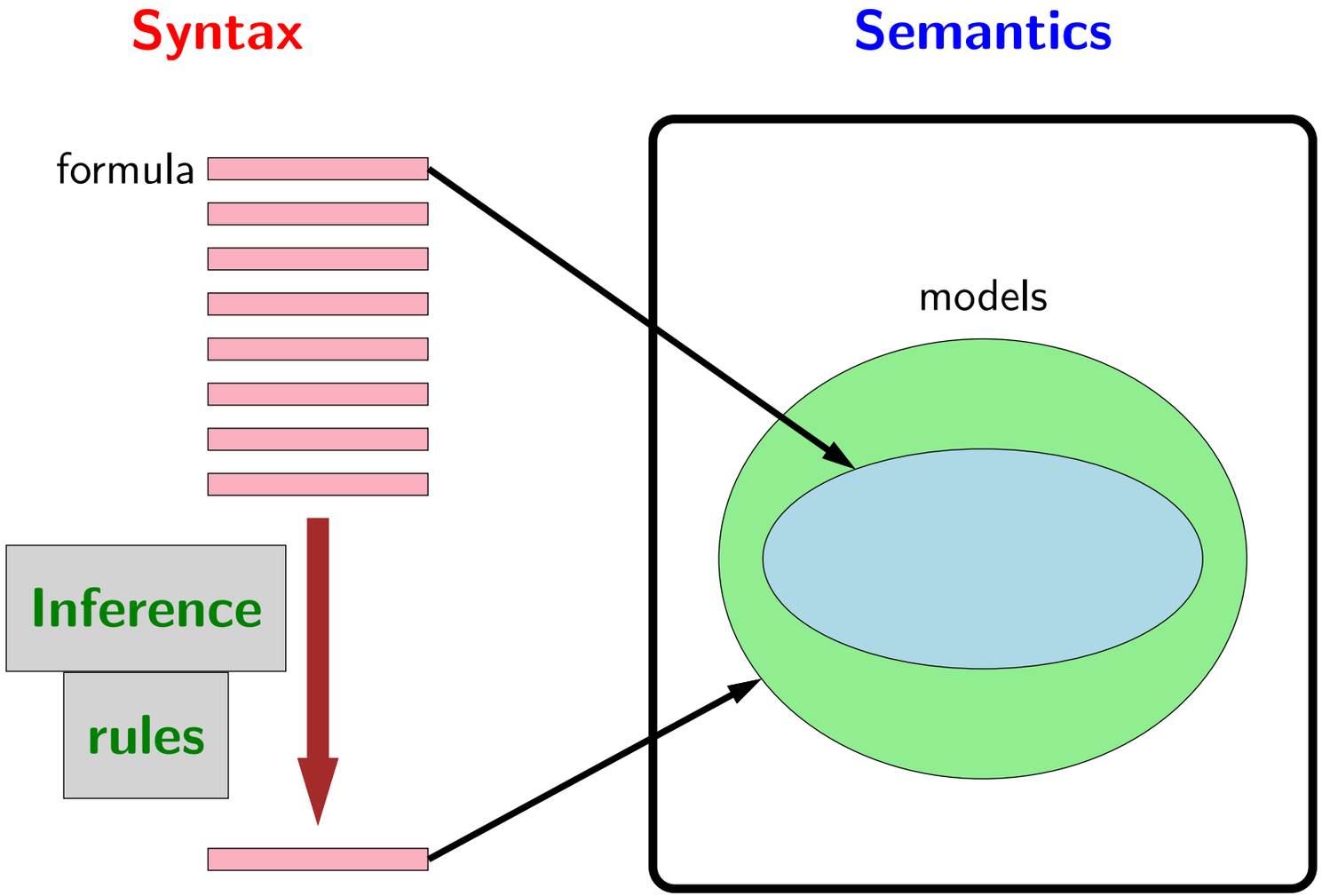
Logic: inference rules

Propositional logic



- So far, we have used formulas, via semantics, to define sets of models. And all our reasoning on formulas has been through these models (e.g., reduction to satisfiability). Inference rules allow us to do reasoning on the formulas themselves without ever instantiating the models.
- This can be quite powerful. If you have a huge KB with lots of formulas and propositional symbols, sometimes you can draw a conclusion without instantiating the full model checking problem. This will be very important when we move to first-order logic, where the models can be infinite, and so model checking would be infeasible.

Inference rules

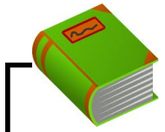
Example of making an inference:

It is raining. (Rain)

If it is raining, then it is wet. (Rain $\rightarrow$ Wet)

Therefore, it is wet. (Wet)

$$\frac{\text{Rain}, \quad \text{Rain} \rightarrow \text{Wet}}{\text{Wet}} \qquad \frac{\text{(premises)}}{\text{(conclusion)}}$$



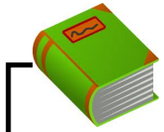
Definition: Modus ponens inference rule

For any propositional symbols p and q:

$$\frac{p, \quad p \rightarrow q}{q}$$

- The idea of making an inference should be quite intuitive to you. The classic example is **modus ponens**, which captures the if-then reasoning pattern.

Inference framework



Definition: inference rule

If $f_1, \ldots, f_k, g$ are formulas, then the following is an **inference rule**:

$$\frac{f_1, \quad \cdots \quad , f_k}{g}$$

Key idea: inference rules

Rules operate directly on **syntax**, not on **semantics**.

- In general, an inference rule has a set of premises and a conclusion. The rule says that if the premises are in the KB, then you can add the conclusion to the KB.
- We haven't yet specified whether this is a valid thing to do, but it is a thing to do. Remember, syntax is just about symbol pushing; it is only by linking to models that we have notions of truth and meaning (semantics).

Inference algorithm

Algorithm: forward inference

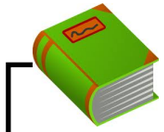
Input: set of inference rules Rules.

Repeat until no changes to KB:

  Choose set of formulas $f_1, \ldots, f_k \in$ KB.

  If matching rule $\frac{f_1, \quad \ldots \quad, f_k}{g}$ exists:

    Add g to KB.



Definition: derivation

KB **derives/proves** f (KB $\vdash f$) iff f eventually gets added to KB.

- Given a set of inference rules (e.g., modus ponens), we can just keep on trying to apply rules. Those rules generate new formulas which get added to the knowledge base, and those formulas might then be premises of other rules, which in turn generate more formulas, etc.
- We say that the KB derives or proves a formula f if by blindly applying rules, we can eventually add f to the KB.

Inference example

Example: Modus ponens inference

Starting point:

$$\text{KB} = \{\text{Rain}, \text{Rain} \rightarrow \text{Wet}, \text{Wet} \rightarrow \text{Slippery}\}$$

Apply modus ponens to Rain and Rain $\rightarrow$ Wet:

$$\text{KB} = \{\text{Rain}, \text{Rain} \rightarrow \text{Wet}, \text{Wet} \rightarrow \text{Slippery}, \text{Wet}\}$$

Apply modus ponens to Wet and Wet $\rightarrow$ Slippery:

$$\text{KB} = \{\text{Rain}, \text{Rain} \rightarrow \text{Wet}, \text{Wet} \rightarrow \text{Slippery}, \text{Wet}, \text{Slippery}\}$$

Converged.

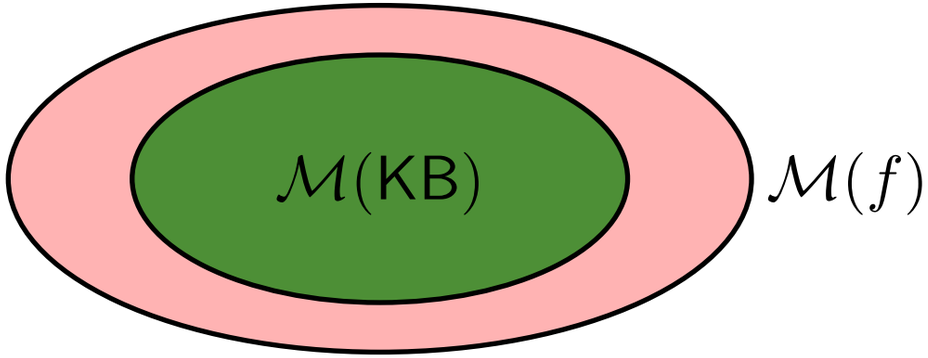
Can't derive some formulas: $\neg\text{Wet}$, $\text{Rain} \rightarrow \text{Slippery}$

- Here is an example where we've applied modus ponens twice. Note that Wet and Slippery are derived by the KB.
- But there are some formulas which cannot be derived. Some of these underivable formulas will look bad anyway ($\neg$Wet), but others will seem reasonable (Rain $\rightarrow$ Slippery).

Desiderata for inference rules

Semantics

Interpretation defines **entailed/true** formulas: $\text{KB} \models f$:



Syntax:

Inference rules **derive** formulas: $\text{KB} \vdash f$

How does $\{f : \text{KB} \models f\}$ relate to $\{f : \text{KB} \vdash f\}$?

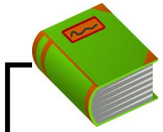
- We can apply inference rules all day long, but now we desperately need some guidance on whether a set of inference rules is doing anything remotely sensible.
- For this, we turn to semantics, which gives an objective notion of truth. Recall that the semantics provides us with $\mathcal{M}$, the set of satisfiable models for each formula f or knowledge base. This defines a set of formulas $\{f : \mathsf{KB} \models f\}$ which are defined to be true.
- On the other hand, inference rules also gives us a mechanism for generating a set of formulas, just by repeated application. This defines another set of formulas $\{f : \mathsf{KB} \vdash f\}$.

Truth

$$\{f : \mathsf{KB} \models f\}$$

- Imagine a glass that represents the set of possible formulas entailed by the KB (these are necessarily true).
- By applying inference rules, we are filling up the glass with water.

Soundness



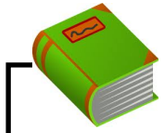
Definition: soundness

A set of inference rules Rules is sound if:

$$\{f : \text{KB} \vdash f\} \subseteq \{f : \text{KB} \models f\}$$

- We say that a set of inference rules is **sound** if using those inference rules, we never overflow the glass: the set of derived formulas is a subset of the set of true/entailed formulas.

Completeness



Definition: completeness

A set of inference rules Rules is complete if:

$$\{f : \text{KB} \vdash f\} \supseteq \{f : \text{KB} \models f\}$$

- We say that a set of inference rules is **complete** if using those inference rules, we fill up the glass to the brim (and possibly go over): the set of derived formulas is a superset of the set of true/entailed formulas.

Soundness and completeness

The truth, the whole truth, and nothing but the truth.

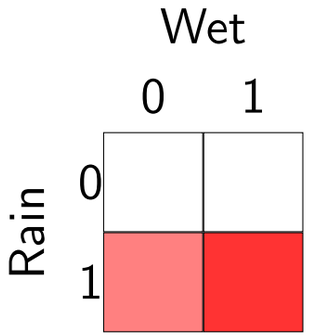
- **Soundness**: nothing but the truth
- **Completeness**: whole truth

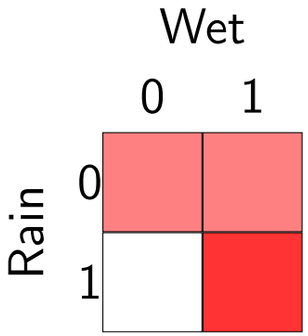
- A slogan to keep in mind is the oath given in a sworn testimony.

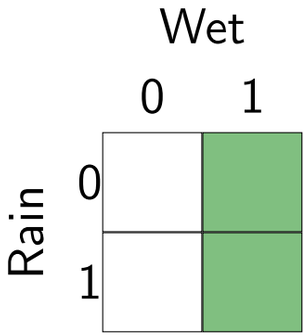
Soundness: example

Is $\dfrac{\text{Rain}, \quad \text{Rain} \to \text{Wet}}{\text{Wet}}$ (Modus ponens) sound?

$\mathcal{M}(\text{Rain}) \quad \cap \quad \mathcal{M}(\text{Rain} \to \text{Wet}) \quad \subseteq ? \quad \mathcal{M}(\text{Wet})$







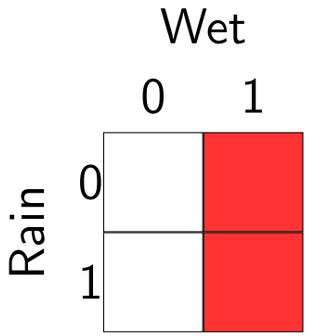
Sound!

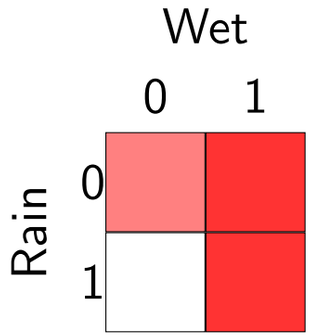
- To check the soundness of a set of rules, it suffices to focus on one rule at a time.
- Take the modus ponens rule, for instance. We can derive Wet using modus ponens. To check entailment, we map all the formulas into semantics-land (the set of satisfiable models). Because the models of Wet is a superset of the intersection of models of Rain and Rain $\rightarrow$ Wet (remember that the models in the KB are an intersection of the models of each formula), we can conclude that Wet is also entailed. If we had other formulas in the KB, that would reduce both sides of $\subseteq$ by the same amount and won't affect the fact that the relation holds. Therefore, this rule is sound.
- Note, we use Wet and Rain to make the example more colorful, but this argument works for arbitrary propositional symbols.

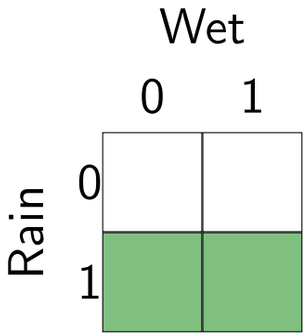
Soundness: example

Is $\dfrac{\text{Wet}, \quad \text{Rain} \rightarrow \text{Wet}}{\text{Rain}}$ sound?

$\mathcal{M}(\text{Wet}) \quad \cap \quad \mathcal{M}(\text{Rain} \rightarrow \text{Wet}) \quad \subseteq? \quad \mathcal{M}(\text{Rain})$







Unsound!

- Here is another example: given Wet and Rain $\rightarrow$ Wet, can we infer Rain? To check it, we mechanically construct the models for the premises and conclusion. Here, the intersection of the models in the premise are not a subset, then the rule is unsound.
- Indeed, backward reasoning is faulty. Note that we can actually do a bit of backward reasoning using Bayesian networks, since we don't have to commit to 0 or 1 for the truth value.

Completeness: example

Recall completeness: inference rules derive all entailed formulas (f such that KB $\models f$)

Example: Modus ponens is incomplete

Setup:

$$\text{KB} = \{\text{Rain}, \text{Rain} \vee \text{Snow} \rightarrow \text{Wet}\}$$

$$f = \text{Wet}$$

$$\text{Rules} = \left\{\frac{f, \quad f \rightarrow g}{g}\right\} \text{ (Modus ponens)}$$

Semantically: KB $\models f$ (f is entailed).

Syntactically: KB $\not\vdash f$ (can't derive f).

Incomplete!

- Completeness is trickier, and here is a simple example that shows that modus ponens alone is not complete, since it can't derive Wet, when semantically, Wet is true!

Fixing completeness

Option 1: Restrict the allowed set of formulas

propositional logic

propositional logic with only Horn clauses

Option 2: Use more powerful inference rules

Modus ponens

resolution

- At this point, there are two ways to fix completeness. First, we can restrict the set of allowed formulas, making the water glass smaller in hopes that modus ponens will be able to fill that smaller glass.
- Second, we can use more powerful inference rules, pouring more vigorously into the same glass in hopes that this will be able to fill the glass; we'll look at one such rule, resolution, in the next lecture.