人工智能技术与应用

逻辑模型

2019.5.27

Question

If $X_1 + X_2 = 10$ and $X_1 - X_2 = 4$, what is X_1?

Some modeling paradigms

State-based models: search problems, MDPs, games

Applications: route finding, game playing, etc.

Think in terms of **states, actions, and costs**

Variable-based models: CSPs, Bayesian networks

Applications: scheduling, tracking, medical diagnosis, etc.

Think in terms of **variables and factors**

Logic-based models: propositional logic, first-order logic

Applications: theorem proving, verification, reasoning

Think in terms of **logical formulas and inference rules**

Taking a step back

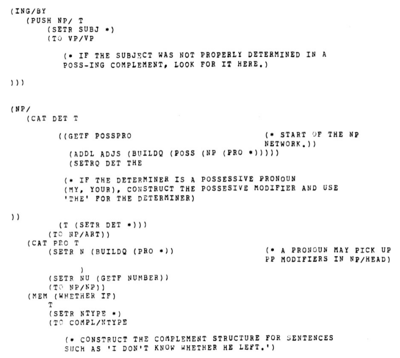
$$\begin{tikzcd}
 & & & \text{query} \arrow[d] \\
\text{data} \arrow[r] & \boxed{\textbf{Learning}} \arrow[r] & \text{model} \arrow[r] & \boxed{\textbf{Inference}} \arrow[d] \\
 & & & \text{prediction}
\end{tikzcd}$$

Examples: search problems, games, neural networks, Bayesian networks

What type of models to use?

A historical note

- Logic was dominant paradigm in AI before 1990s



```
(ING/BY
   (PUSH NP/ T
        (SETR SUBJ *)
        (TO VP/VP

            (* IF THE SUBJECT WAS NOT PROPERLY DETERMINED IN A
            POSS-ING COMPLEMENT, LOOK FOR IT HERE.)

)))

(NP/
   (CAT DET T

           ((GETF POSSPRO                          (* START OF THE NP
                                                   NETWORK.))
             (ADDL ADJS (BUILDQ (POSS (NP (PRO *)))))
             (SETRQ DET THE

            (* IF THE DETERMINER IS A POSSESSIVE PRONOUN
            (MY, YOUR), CONSTRUCT THE POSSESIVE MODIFIER AND USE
            'THE' FOR THE DETERMINER)

))
           (T (SETR DET *)))
        (TO NP/ART))
   (CAT PRO T
        (SETR N (BUILDQ (PRO *))                   (* A PRONOUN MAY PICK UP
                                                   PP MODIFIERS IN NP/HEAD)
              )
        (SETR NU (GETF NUMBER))
        (TO NP/NP))
   (MEM (WHETHER IF)
        T
        (SETR NTYPE *)
        (TO COMPL/NTYPE

            (* CONSTRUCT THE COMPLEMENT STRUCTURE FOR SENTENCES
            SUCH AS 'I DON'T KNOW WHETHER HE LEFT.')
```

- Problem 1: deterministic, didn’t handle **uncertainty** (probability addresses this)
- Problem 2: rule-based, didn’t allow fine tuning from **data** (machine learning addresses this)
- Strength: provides **expressiveness** in a compact way

Motivation: smart personal assistant

Motivation: smart personal assistant

Tell information → ← **Ask** questions

Use natural language!

Need to:

- Digest **heterogenous** information
- Reason **deeply** with that information

Natural language

Example:

- A **dime** is better than a **nickel**.
- A **nickel** is better than a **penny**.
- Therefore, a **dime** is better than a **penny**.

Example:

- A **penny** is better than **nothing**.
- **Nothing** is better than **world peace**.
- Therefore, a **penny** is better than **world peace**???

Natural language is slippery...

Language

Language *is a mechanism for expression.*

Natural languages (informal):

English: *Two divides even numbers.*

German: *Zwei dividieren geraden zahlen.*

Programming languages (formal):

Python: `def even(x): return x % 2 == 0`

C++: `bool even(int x) {return x % 2 == 0; }`

Logical languages (formal):

First-order-logic: $\forall x. \mathrm{Even}(x) \rightarrow \mathrm{Divides}(x, 2)$

Two goals of a logic language

- **Represent** knowledge about the world

- **Reason** with that knowledge

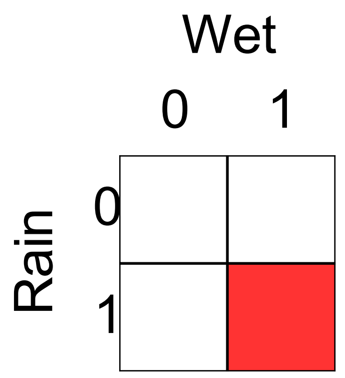
Ingredients of a logic

Syntax: defines a set of valid **formulas** (Formulas)

Example: Rain $\wedge$ Wet

Semantics: for each formula, specify a set of **models** (assignments / configurations of the world)

Example:



Inference rules: given f, what new formulas g can be added that are guaranteed to follow ($\frac{f}{g}$)?

Example: from Rain $\wedge$ Wet, derive Rain

Syntax versus semantics

Syntax: what are valid expressions in the language?

Semantics: what do these expressions mean?

Different syntax, same semantics (5):

`2 + 3` $\Leftrightarrow$ `3 + 2`

Same syntax, different semantics (1 versus 1.5):

`3 / 2` (Python 2.7) $\not\Leftrightarrow$ `3 / 2` (Python 3)

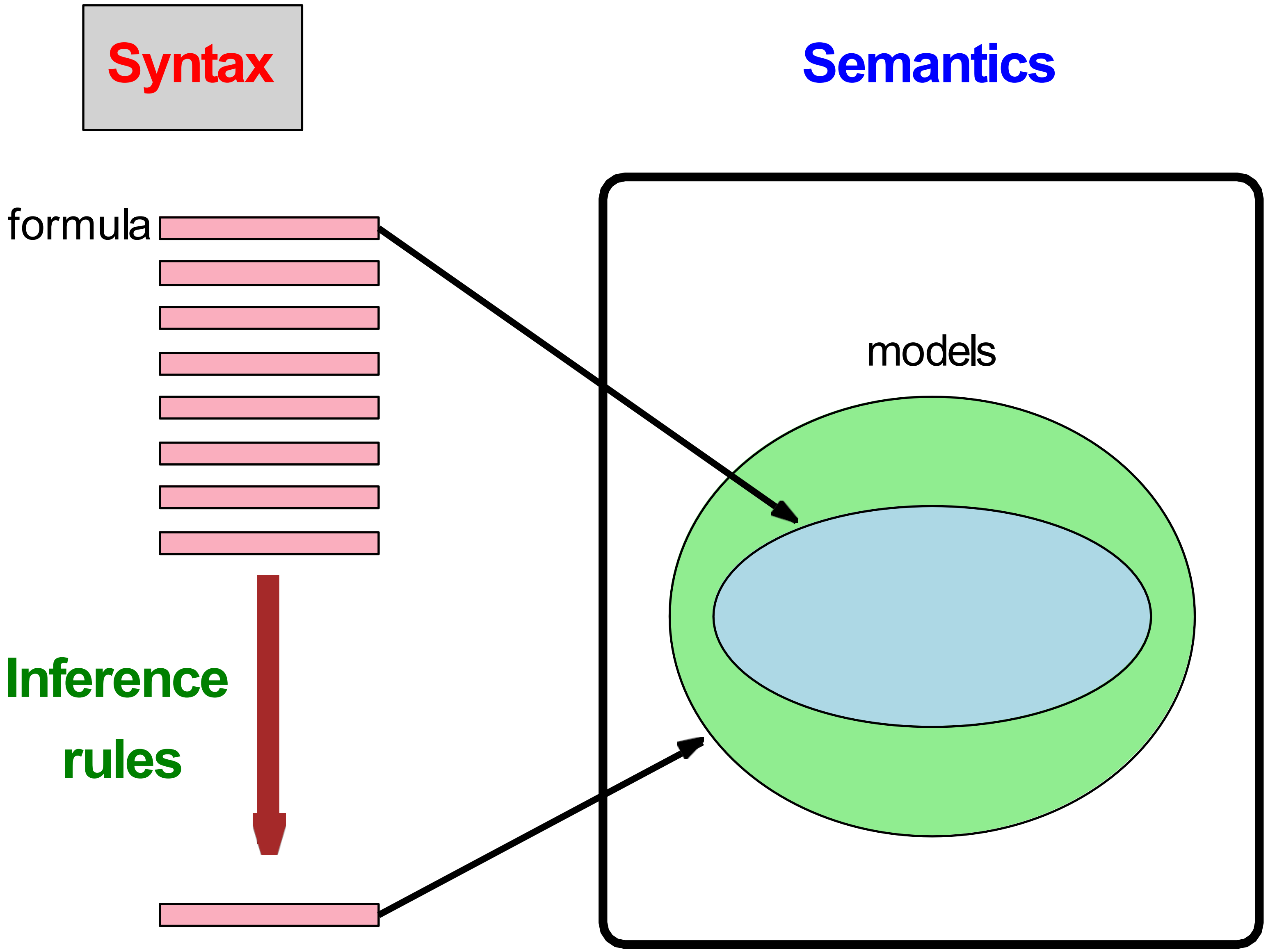
Logics

- **Propositional logic with only Horn clauses**
- **Propositional logic**
- Modal logic
- **First-order logic with only Horn clauses**
- **First-order logic**
- Second-order logic
- ...

Key idea: tradeoff

Balance **expressivity** and **computational efficiency**.

Propositional logic



Syntax of propositional logic

Propositional symbols (atomic formulas): A, B, C

Logical connectives: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$

Build up formulas recursively—if f and g are formulas, so are the following:

- Negation: $\neg f$
- Conjunction: $f \wedge g$
- Disjunction: $f \vee g$
- Implication: $f \rightarrow g$
- Biconditional: $f \leftrightarrow g$

Syntax of propositional logic

- Formula: A
- Formula: $\neg A$
- Formula: $\neg B \rightarrow C$
- Formula: $\neg A \wedge (\neg B \rightarrow C) \vee (\neg B \vee D)$
- Formula: $\neg\neg A$
- Non-formula: $A \neg B$
- Non-formula: $A + B$

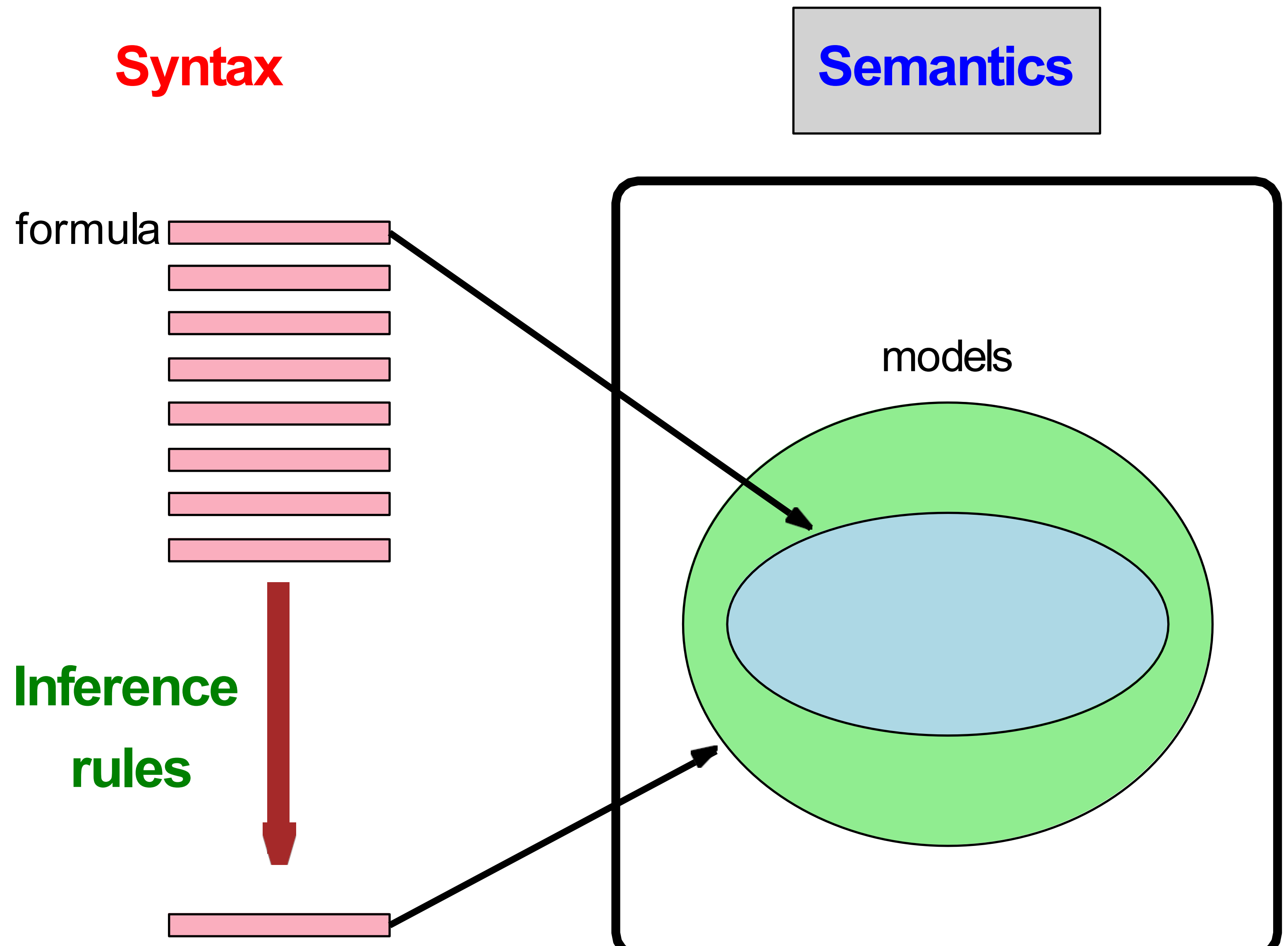
Syntax of propositional logic

Key idea: syntax provides symbols

Formulas by themselves are just symbols (syntax).

No meaning yet (semantics)!

Propositional logic



Model

Definition: model

A **model** w in propositional logic is an **assignment** of truth values to propositional symbols.

Example:

- 3 propositional symbols: A, B, C
- $2^3 =$ 8 possible models w:

$$\{A:0, B:0, C:0\}$$
$$\{A:0, B:0, C:1\}$$
$$\{A:0, B:1, C:0\}$$
$$\{A:0, B:1, C:1\}$$
$$\{A:1, B:0, C:0\}$$
$$\{A:1, B:0, C:1\}$$
$$\{A:1, B:1, C:0\}$$
$$\{A:1, B:1, C:1\}$$

Interpretation function

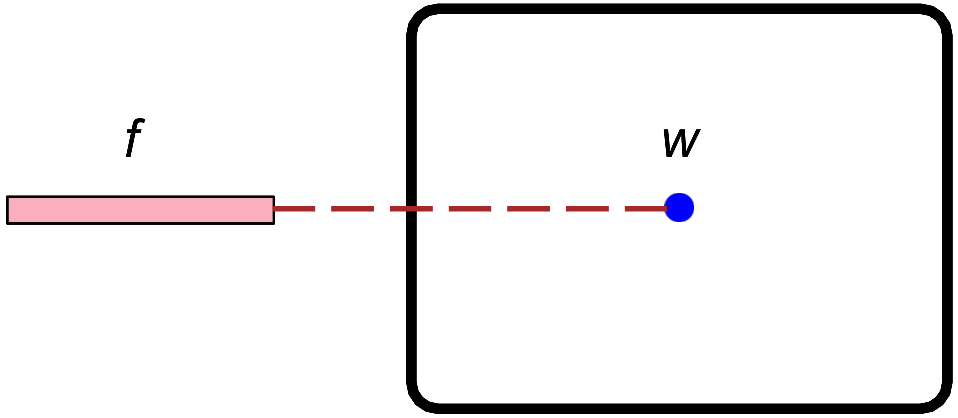
Definition: interpretation function

Let f be a formula.

Let w be a model.

An **interpretation function** $\mathrm{I}(f, w)$ returns:

- true (1) (say that w satisfies f)
- false (0) (say that w does not satisfy f)



Interpretation function: definition

Base case:

- For a propositional symbol p (e.g., A, B, C): $\mathrm{I}(p, w) = w(p)$

Recursive case:

- For any two formulas f and g, define:

$\mathrm{I}(f, w)$	$\mathrm{I}(g, w)$	$\mathrm{I}(\neg f, w)$	$\mathrm{I}(f \wedge g, w)$	$\mathrm{I}(f \vee g, w)$	$\mathrm{I}(f \rightarrow g, w)$	$\mathrm{I}(f \leftrightarrow g, w)$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Interpretation function: example

Example: interpretation function

Formula: $f = (\neg A \wedge B) \leftrightarrow C$

Model: $w = \{A : 1, B : 1, C : 0\}$

Interpretation:

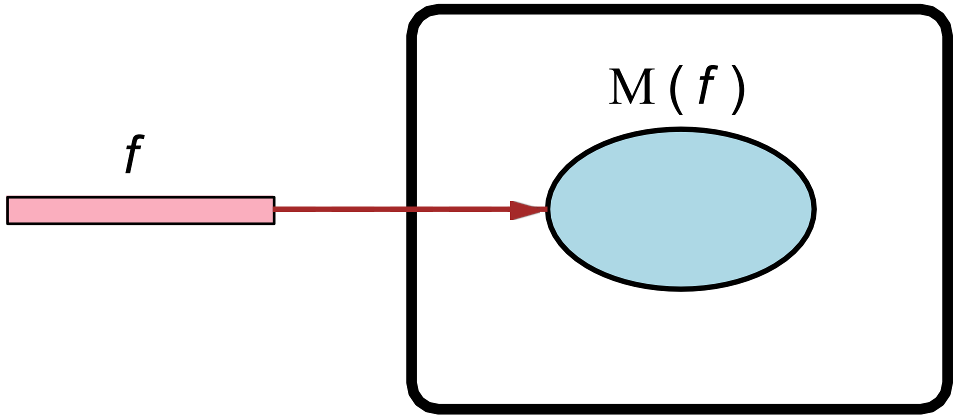
$$\begin{tikzcd}
 & & I((\neg A \wedge B) \leftrightarrow C, w) = 1 \arrow[dl, dash] \arrow[dr, dash] & \\
 & \boxed{I(\neg A \wedge B, w) = 0} \arrow[dl, dash] \arrow[dr, dash] & & \boxed{I(C, w) = 0} \\
\boxed{I(\neg A, w) = 0} \arrow[d, dash] & & \boxed{I(B, w) = 1} & \\
\boxed{I(A, w) = 1} & & &
\end{tikzcd}$$

Formula represents a set of models

So far: each formula f and model w has an interpretation $\mathcal{I}(f, w) \in \{0, 1\}$

Definition: models

Let $\mathcal{M}(f)$ be the set of **models** w for which $\mathcal{I}(f, w) = 1$.



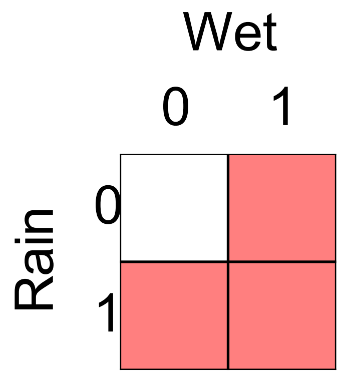
Models: example

Formula:

$$f = \text{Rain} \vee \text{Wet}$$

Models:

$$M(f) =$$



Key idea: compact representation

A **formula** *compactly* represents a set of **models**.

Knowledge base

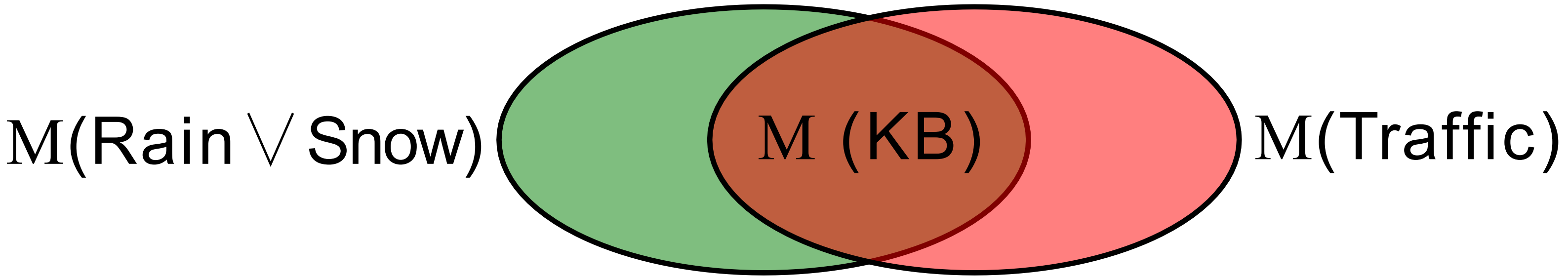
Definition: Knowledge base

A **knowledge base** KB is a set of formulas representing their conjunction / intersection:

$$\mathcal{M}(\mathrm{KB}) = \bigcap_{f \in \mathrm{KB}} \mathcal{M}(f).$$

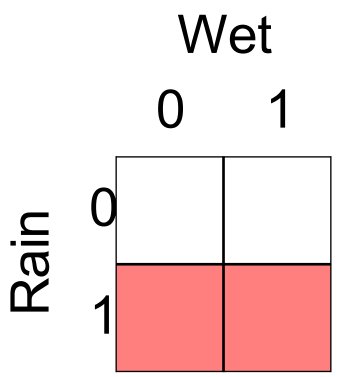
Intuition: KB specifies constraints on the world. $\mathcal{M}(\mathrm{KB})$ is the set of all worlds satisfying those constraints.

Let KB = $\{\text{Rain} \vee \text{Snow}, \text{Traffic}\}$.

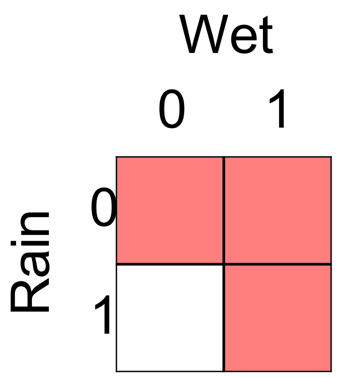


Knowledge base: example

M (Rain)

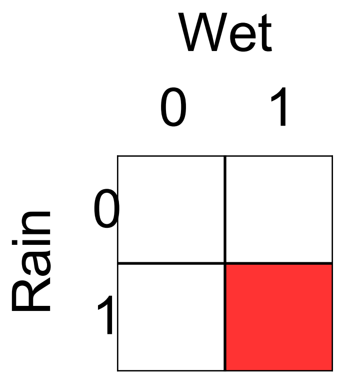


$\mathrm{M}(\text{Rain} \rightarrow \text{Wet})$



Intersection:

$\mathrm{M}(\{\text{Rain, Rain} \rightarrow \text{Wet}\})$



Adding to the knowledge base

Adding more formulas to the knowledge base:

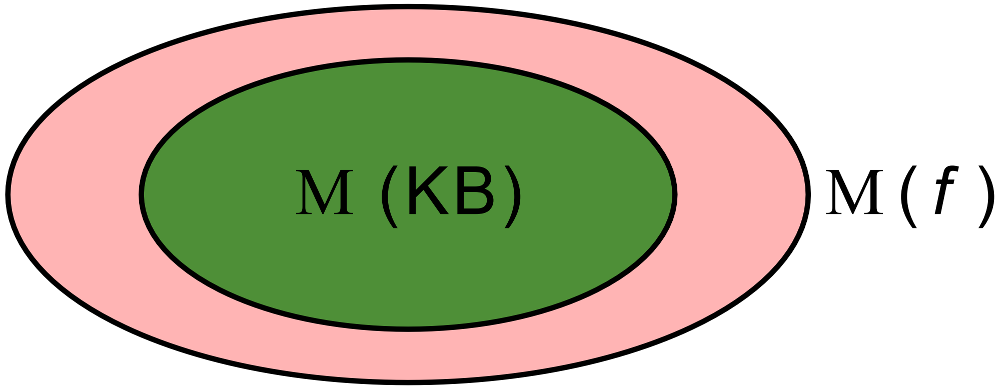
$$\mathrm{KB} \longrightarrow \mathrm{KB} \cup \{f\}$$

Shrinks the set of models:

$$\mathrm{M}(\mathrm{KB}) \longrightarrow \mathrm{M}(\mathrm{KB}) \cap \mathrm{M}(f)$$

How much does $\mathrm{M}(\mathrm{KB})$ shrink?

Entailment (蕴涵)



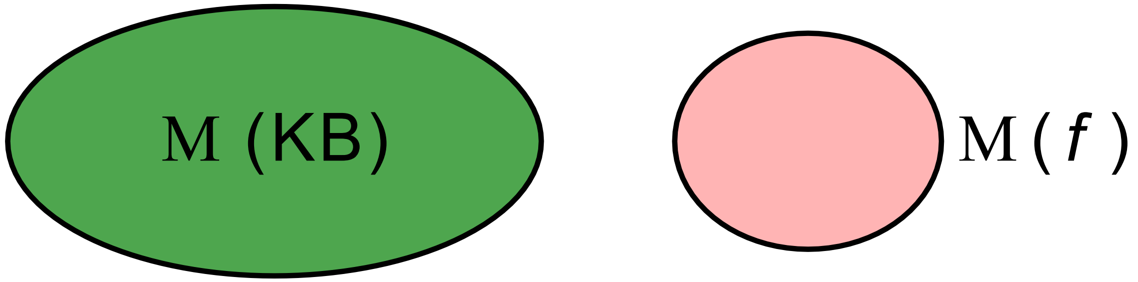
Intuition: *f* added no information/constraints (it was already known).

Definition: entailment

KB entails *f* (written KB $\models f$) iff $M(f) \supseteq M(KB)$.

Example: Rain $\wedge$ Snow $\models$ Snow

Contradiction



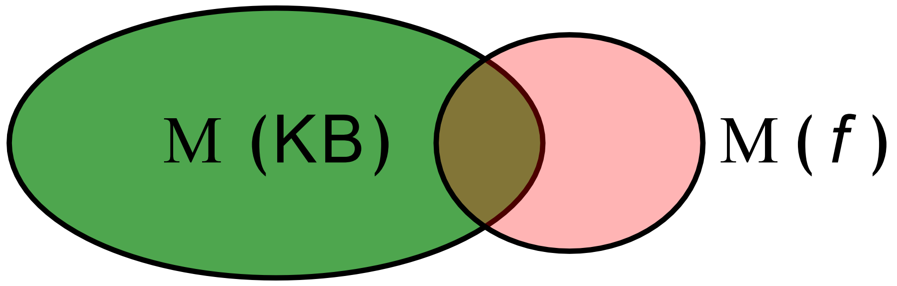
Intuition: f contradicts what we know (captured in KB).

Definition: contradiction

KB contradicts f iff $\mathcal{M}(\mathrm{KB}) \cap \mathcal{M}(f) = \varnothing$.

Example: Rain $\wedge$ Snow contradicts $\neg$Snow

Contingency



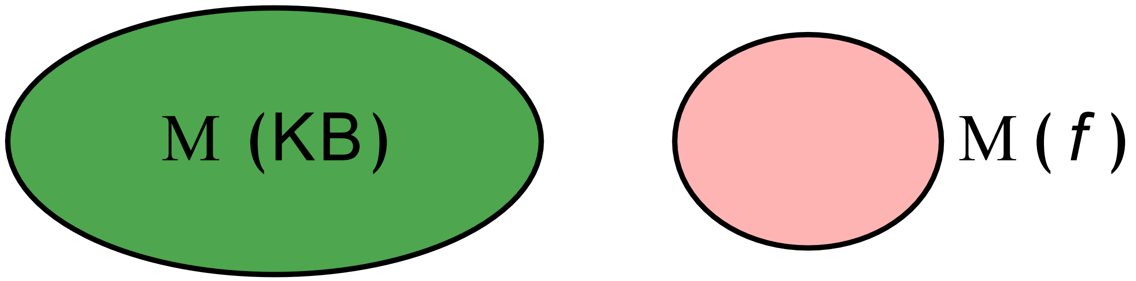
Intuition: *f* adds non-trivial information to KB

$$\emptyset \subset M(KB) \cap M(f) \subset M(KB)$$

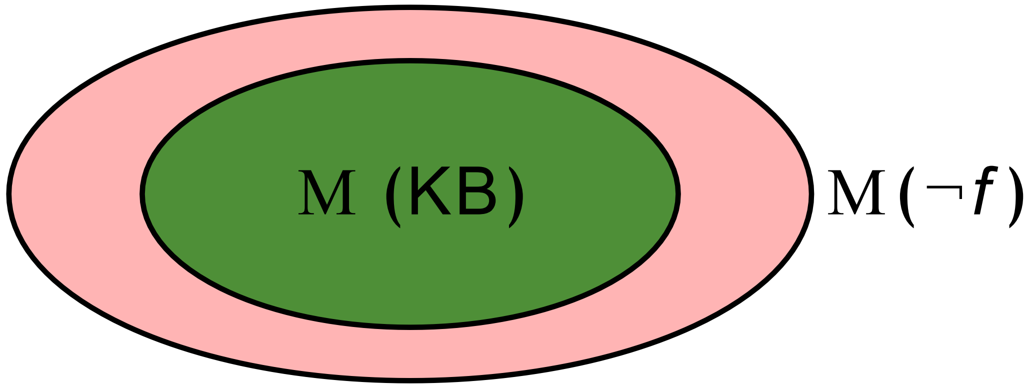
Example: Rain and Snow

Contradiction and entailment

Contradiction:



Entailment:



Proposition: contradiction and entailment

KB **contradicts** *f* iff KB **entails** ¬*f* .

Tell operation

$$\text{Tell}[f] \longrightarrow \text{KB} \longrightarrow ?$$

Tell: *It is raining.*

Tell[Rain]

Possible responses:

- Already knew that: entailment (KB |= *f*)
- Don't believe that: contradiction (KB |= ¬*f*)
- Learned something new (update KB): contingent

Ask operation

$$\text{Ask}[f] \longrightarrow \text{KB} \longrightarrow ?$$

Ask: *Is it raining?*

Ask[Rain]

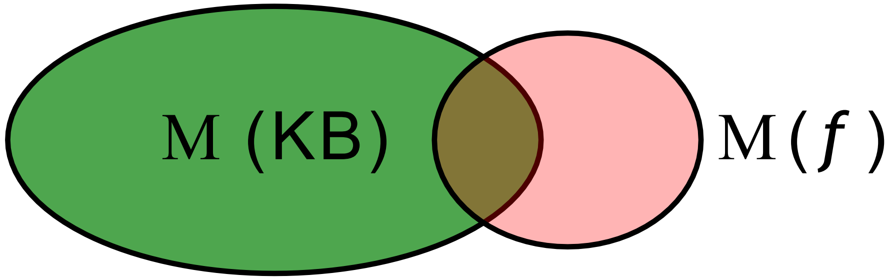
Possible responses:

- Yes: entailment (KB $\models f$)
- No: contradiction (KB $\models \neg f$)
- I don't know: contingent

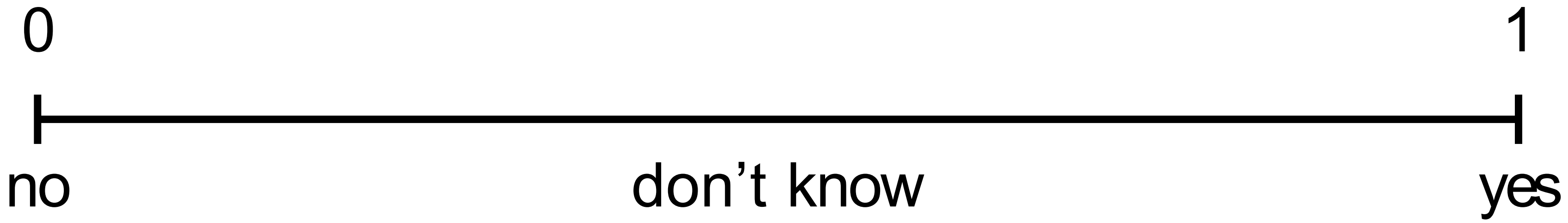
Digression: probabilistic generalization

Bayesian network: distribution over assignments (models)

w	$P(W = w)$
{A: 0, B: 0, C: 0 }	0.3
{A: 0, B: 0, C: 1 }	0.1
...	...



$$P(f \mid \mathrm{KB}) = \frac{\sum_{w \in \mathcal{M}(\mathrm{KB} \cup \{f\})} P(W = w)}{\sum_{w \in \mathcal{M}(\mathrm{KB})} P(W = w)}$$

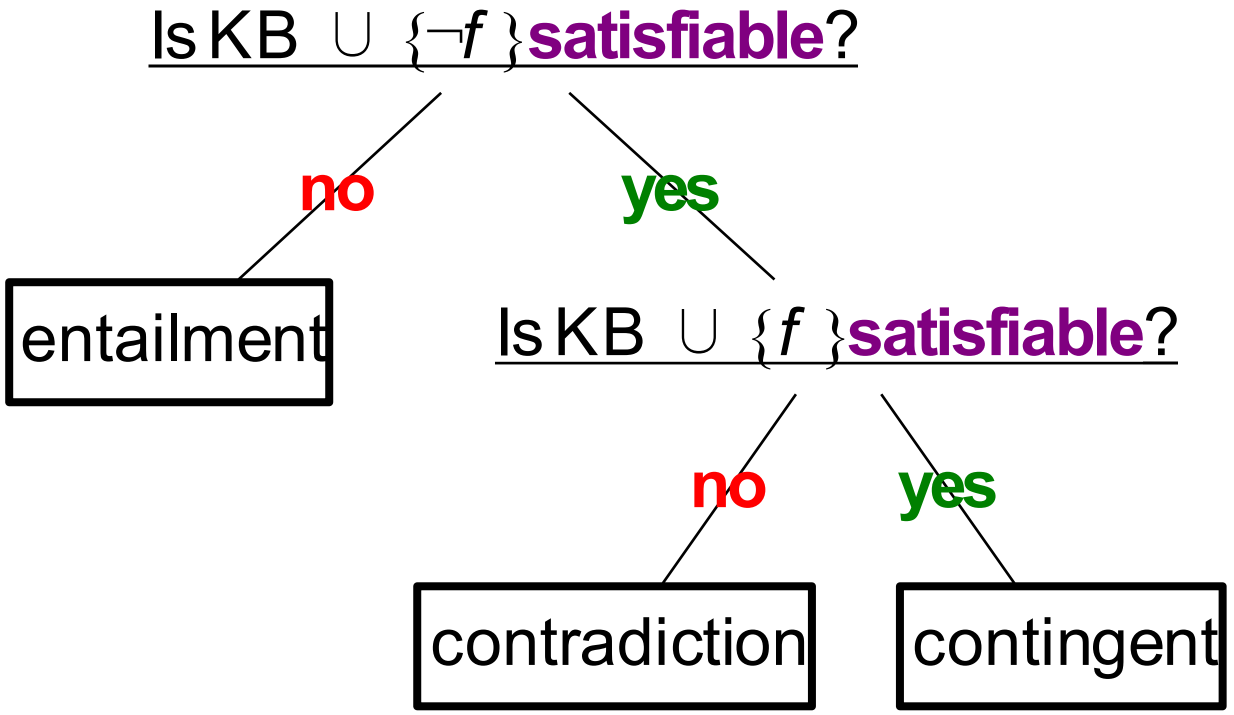


Satisfiability

Definition: satisfiability

A knowledge base KB is **satisfiable** if $\mathcal{M}(KB) \neq \varnothing$.

Reduce Ask[f] and Tell[f] to satisfiability:


Is KB $\cup \{\neg f\}$ **satisfiable**?
no
yes
entailment
Is KB $\cup \{f\}$ **satisfiable**?
no
yes
contradiction
contingent

Model checking

Checking satisfiability (SAT) in propositional logic is special case of solving CSPs!

Mapping:

propositional symbol	$\Rightarrow$	variable
formula	$\Rightarrow$	constraint
model	$\Leftarrow$	assignment

Model checking

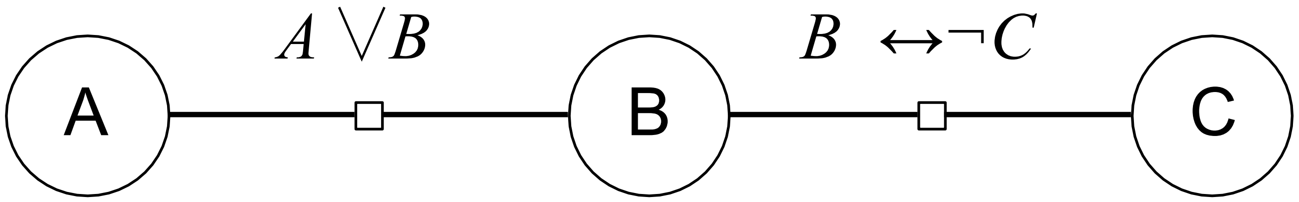
Example: model checking

KB = $\{A \vee B,\ B \leftrightarrow \neg C\}$

Propositional symbols (CSP variables):

$$\{A, B, C\}$$

CSP:



Consistent assignment (satisfying model):

$$\{A : 1, B : 0, C : 1\}$$

Model checking

Definition: model checking

Input: knowledge base KB

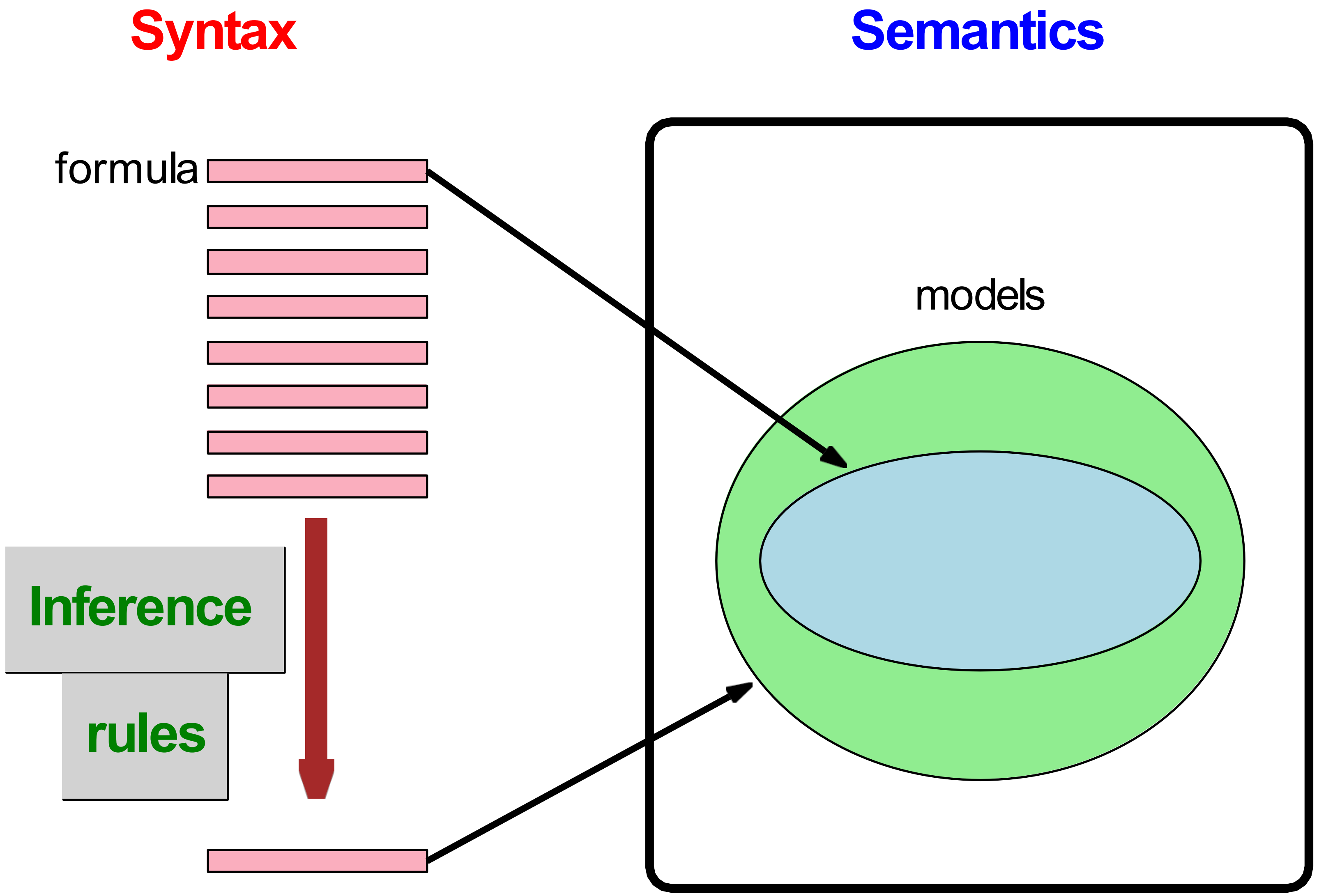
Output: exists satisfying model ($\mathcal{M}(\text{KB}) \neq \varnothing$)?

Popular algorithms:

- DPLL (backtracking search + pruning)
- WalkSat (randomized local search)

Next: Can we exploit the fact that factors are formulas?

Propositional logic



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*:

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

Inference framework

Definition: inference rule

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

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

Key idea: inference rules

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

Inference algorithm

Algorithm: forward inference

```
Input: set of inference rules Rules.
Repeat until no changes to KB:
    Choose set of formulas $f_1, \dots, f_k \in$ KB
    If matching rule $\frac{f_1, \quad \dots \quad, f_k}{g}$ exists:
        Add $g$ to KB.
```

Definition: derivation

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

Inference example

Example: Modus ponens inference

Starting point:

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

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

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

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

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

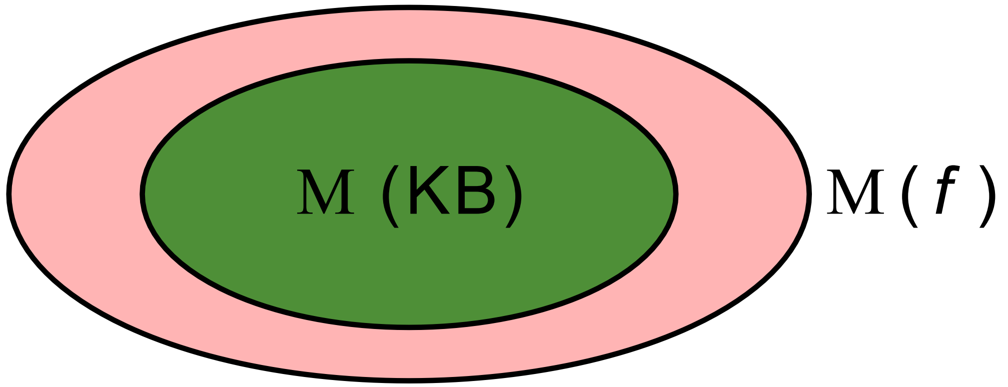
Converged.

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

Desidarata for inference rules

Semantics

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



Syntax:

Inference rules **derive** formulas: $KB \vdash f$

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

Truth

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

Soundness

Definition: soundness

A set of inference rules Rules is sound if:

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

Completeness

Definition: completeness

A set of inference rules Rules is complete if:

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

Soundness and completeness

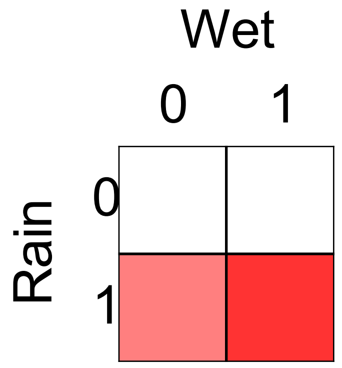
The truth, the whole truth, and nothing but the truth.

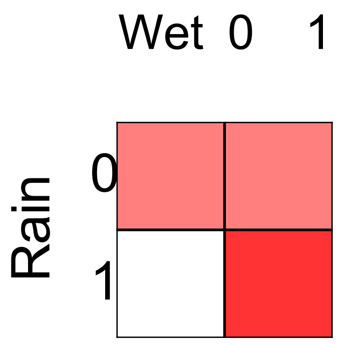
- **Soundness**: nothing but the truth
- **Completeness**: whole truth

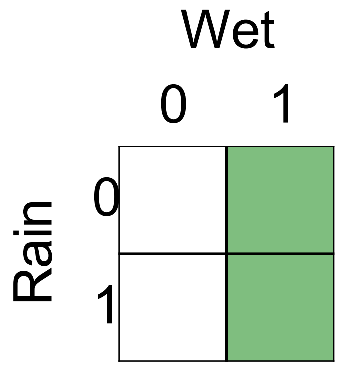
Soundness: example

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

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





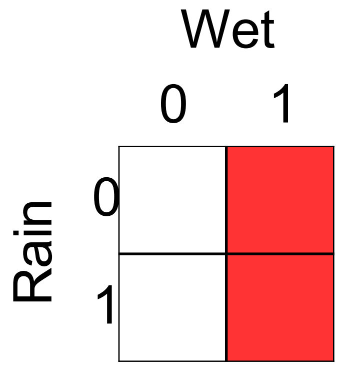


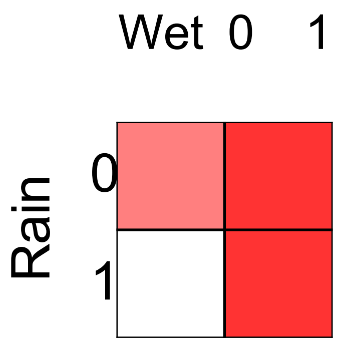
Sound!

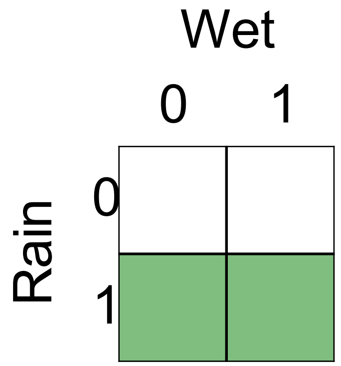
Soundness: example

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

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







Unsound!

Completeness: example

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

Example: Modus ponens is incomplete

Setup:

KB = $\{\text{Rain}, \text{Rain} \lor \text{Snow} \to \text{Wet}\}$

f = Wet

Rules = $\{ \frac{f, \quad f \to g}{g} \}$ (Modus ponens)

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

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

Incomplete!

Fixing completeness

Option 1: Restrict the allowed set of formulas

propositional logic

$\downarrow$

propositional logic with only Horn clauses

Option 2: Use more powerful inference rules

Modus ponens

$\downarrow$

resolution

Definite clauses

Definition: Definite clause

A **definite clause** has the following form:

$$(p_1 \wedge \cdots \wedge p_k) \rightarrow q$$

where $p_1, \ldots, p_k, q$ are propositional symbols.

Intuition: if $p_1, \ldots, p_k$ hold, then q holds.

Example: (Rain $\wedge$ Snow) $\rightarrow$ Traffic

Example: Traffic

Non-example: Rain $\wedge$ Snow Non-

example: $\neg$Traffic

Non-example: (Rain $\wedge$ Snow) $\rightarrow$ (Traffic $\vee$ Peaceful)

Horn clauses

Definition: Horn clause

A **Horn clause** is either:

- a definite clause ($p_1 \wedge \cdots \wedge p_k \rightarrow q$)
- a goal clause ($p_1 \wedge \cdots \wedge p_k \rightarrow \text{false}$)

Example (definite): $(\text{Rain} \wedge \text{Snow}) \rightarrow \text{Traffic}$

Example (goal): $\text{Traffic} \wedge \text{Accident} \rightarrow \text{false}$

equivalent: $\neg(\text{Traffic} \wedge \text{Accident})$

Modus ponens

Inference rule:

Definition: Modus ponens

$$\frac{p_1, \quad \cdots \quad , p_k, \quad (p_1 \wedge \cdots \wedge p_k) \rightarrow q}{q}$$

Example:

Example: Modus ponens

$$\frac{\text{Wet}, \quad \text{Weekday}, \quad \text{Wet} \wedge \text{Weekday} \rightarrow \text{Traffic}}{\text{Traffic}}$$

Completeness of modus ponens

Theorem: Modus ponens on Horn clauses

Modus ponens is **complete** with respect to Horn clauses:

- Suppose KB contains only Horn clauses and p is an entailed propositional symbol.
- Then applying modus ponens will derive p.

Upshot:

KB $\models p$ (entailment) is the same as KB $\vdash p$ (derivation)!

Answering questions

KB

Rain Weekday

Rain $\to$ Wet

Wet $\wedge$ Weekday $\to$ Traffic

Traffic $\wedge$ Careless $\to$ Accident

Definition: Modus ponens

$$\frac{p_1, \ \ldots, \ p_k, \quad (p_1 \wedge \cdots \wedge p_k) \to q}{q}$$

Query: Ask[Traffic]

"Yes" subproblem: KB $\vDash$ Traffic

Equivalent: KB contradicts $\neg$Traffic

Equivalent: KB $\cup$ $\{$Traffic $\to$ false$\}$ $\vdash$ false?

"No" subproblem: KB $\vDash$ $\neg$Traffic

Equivalent: KB $\vdash$ $\neg$Traffic — **impossible!**

"Yes" subproblem

KB

Rain

Weekday

Rain $\rightarrow$ Wet

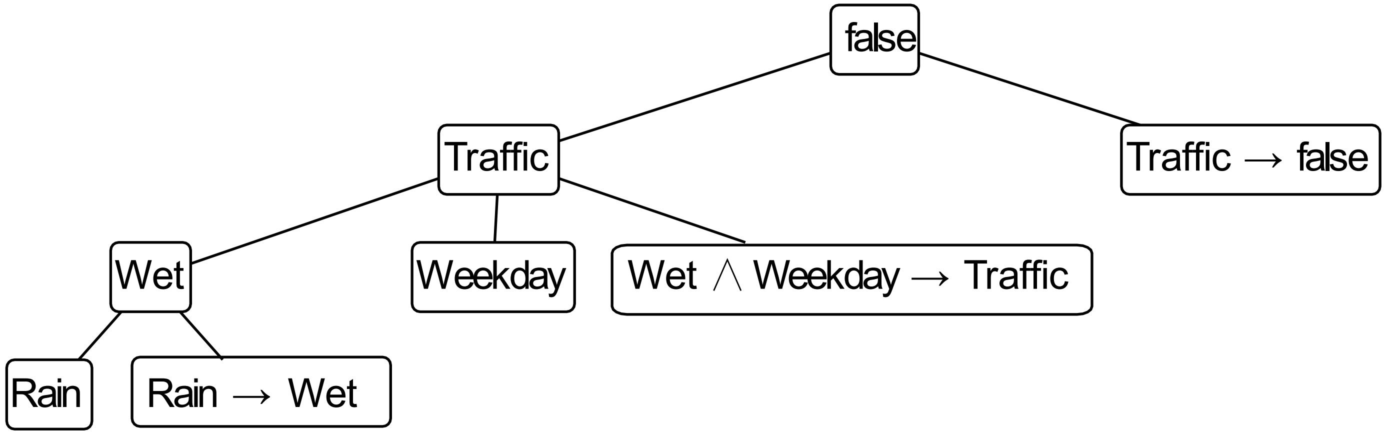
Wet $\wedge$ Weekday $\rightarrow$ Traffic

Traffic $\wedge$ Careless $\rightarrow$ Accident

Definition: Modus ponens

$$\frac{p_1, \cdots, p_k, \quad (p_1 \wedge \cdots \wedge p_k) \rightarrow q}{q}$$

Question: KB $\cup$ {Traffic $\rightarrow$ false} $\vdash$ false?



Approaches

Formulas allowed	Inference rule	Complete?
Propositional logic (only Horn clauses)	modus ponens	yes
Propositional logic	modus ponens	no
Propositional logic	**resolution**	yes

Horn clauses and disjunction

Written with implication	**Written with disjunction**
$A \rightarrow C$	$\neg A \vee C$
$A \wedge B \rightarrow C$	$\neg A \vee \neg B \vee C$

- **Literal**: either p or $\neg p$, where p is a propositional symbol
- **Clause**: disjunction of literals
- **Horn clauses**: at most one positive literal

Modus ponens (rewritten):

$$\frac{A, \quad \neg A \vee C}{C}$$

- Intuition: cancel out A and $\neg A$

Resolution [Robinson, 1965]

General clauses have any number of literals:

$$\neg A \vee B \vee \neg C \vee D \vee \neg E \vee F$$

Example: resolution inference rule

$$\frac{\text{Rain} \vee \text{Snow}, \quad \neg\text{Snow} \vee \text{Traffic}}{\text{Rain} \vee \text{Traffic}}$$

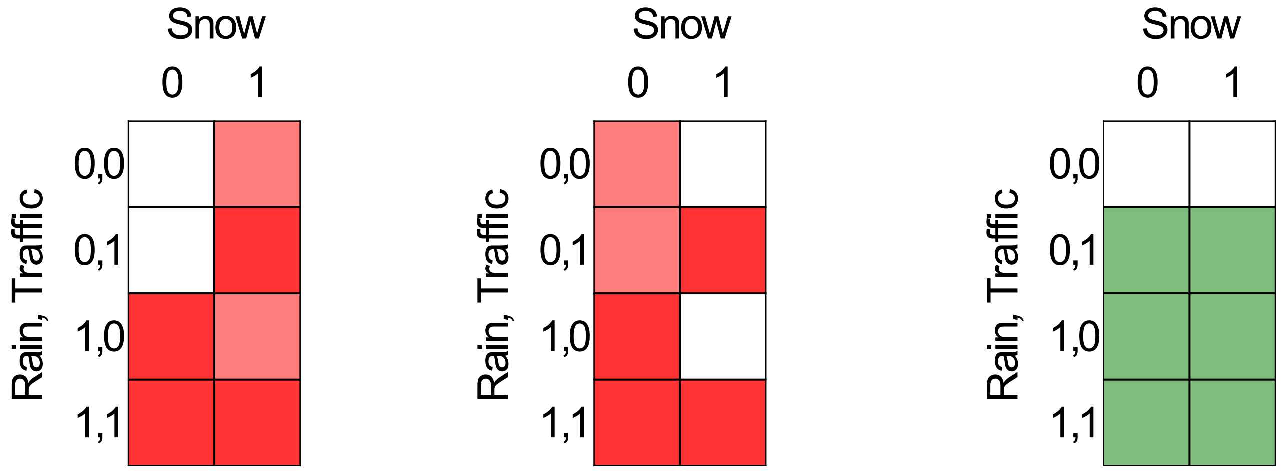
Definition: resolution inference rule

$$\frac{f_1 \vee \cdots \vee f_n \vee p, \quad \neg p \vee g_1 \vee \cdots \vee g_m}{f_1 \vee \cdots \vee f_n \vee g_1 \vee \cdots \vee g_m}$$

Soundness of resolution

$$\frac{\text{Rain} \lor \text{Snow}, \quad \neg\text{Snow} \lor \text{Traffic}}{\text{Rain} \lor \text{Traffic}} \quad \text{(resolution rule)}$$

$M(\text{Rain} \lor \text{Snow}) \cap M(\neg\text{Snow} \lor \text{Traffic}) \subseteq ?\, M(\text{Rain} \lor \text{Traffic})$



Sound!

Conjunctive normal form

So far: resolution only works on clauses...but that's enough!

Definition: conjunctive normal form (CNF)

A **CNF formula** is a conjunction of clauses.

Example: $(A \vee B \vee \neg C) \wedge (\neg B \vee D)$

Equivalent: knowledge base where each formula is a clause

Proposition: conversion to CNF

Every formula f in propositional logic can be converted into an equivalent CNF formula f':

$$\mathcal{M}(f) = \mathcal{M}(f')$$

Conversion to CNF: example

Initial formula:

$$(\text{Summer} \rightarrow \text{Snow}) \rightarrow \text{Bizzare}$$

Remove implication ($\rightarrow$):

$$\neg(\neg\text{Summer} \vee \text{Snow}) \vee \text{Bizzare}$$

Push negation ($\neg$) inwards (de Morgan):

$$(\neg\neg\text{Summer} \wedge \neg\text{Snow}) \vee \text{Bizzare}$$

Remove double negation:

$$(\text{Summer} \wedge \neg\text{Snow}) \vee \text{Bizzare}$$

Distribute $\vee$ over $\wedge$:

$$(\text{Summer} \vee \text{Bizzare}) \wedge (\neg\text{Snow} \vee \text{Bizzare})$$

Conversion to CNF: general

Conversion rules:

- Eliminate $\leftrightarrow$: $\dfrac{f \leftrightarrow g}{(f \rightarrow g) \wedge (g \rightarrow f)}$
- Eliminate $\rightarrow$: $\dfrac{f \rightarrow g}{\neg f \vee g}$
- Move $\neg$ inwards: $\dfrac{\neg(f \wedge g)}{\neg f \vee \neg g}$
- Move $\neg$ inwards: $\dfrac{\neg(f \vee g)}{\neg f \wedge \neg g}$
- Eliminate double negation: $\dfrac{\neg\neg f}{f}$
- Distribute $\vee$ over $\wedge$: $\dfrac{f \vee (g \wedge h)}{(f \vee g) \wedge (f \vee h)}$

Resolution algorithm

Recall: entailment and contradiction $\Leftrightarrow$ satisfiability

$\text{KB} \models f \quad \Longleftrightarrow \quad \text{KB} \cup \{\neg f\}$ is unsatisfiable

$\text{KB} \models \neg f \quad \Longleftrightarrow \quad \text{KB} \cup \{f\}$ is unsatisfiable

Algorithm: resolution-based inference

- Convert all formulas into **CNF**.
- Repeatedly apply **resolution** rule.
- Return unsatisfiable iff derive false.

Resolution: example

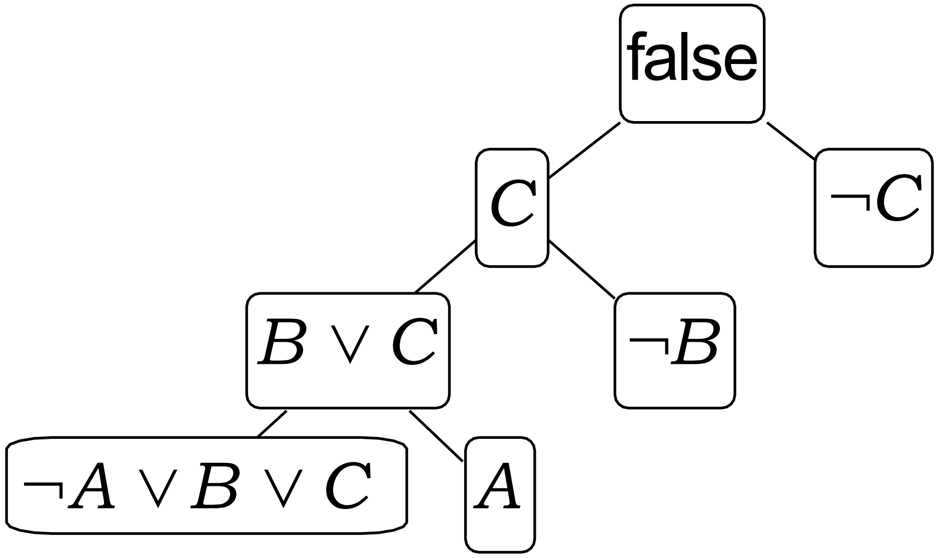
Knowledge base (is it satisfiable?): KB

$$= \{A \rightarrow (B \vee C), A, \neg B, \neg C\}$$

Convert to CNF:

KB $= \{\neg A \vee B \vee C, A, \neg B, \neg C\}$

Repeatedly apply **resolution** rule:



Unsatisfiable!

Time complexity

Definition: modus ponens inference rule

$$\frac{p_1, \quad \cdots \quad , p_k, \quad (p_1 \wedge \cdots \wedge p_k) \rightarrow q}{q}$$

- Each rule application adds clause with **one** propositional symbol
  ⇒ linear time

Definition: resolution inference rule

$$\frac{f_1 \vee \cdots \vee f_n \vee p, \quad \neg p \vee g_1 \vee \cdots \vee g_m}{f_1 \vee \cdots \vee f_n \vee g_1 \vee \cdots \vee g_m}$$

- Each rule application adds clause with **many** propositional symbols
  ⇒ exponential time

Summary

Horn clauses	any clauses
modus ponens	resolution
linear time	exponential time
less expressive	more expressive

Limitations of propositional logic

Alice and Bob both know arithmetic.

AliceKnowsArithmetic ∧ BobKnowsArithmetic

All students know arithmetic.

AliceIsStudent → AliceKnowsArithmetic

BobIsStudent → BobKnowsArithmetic

. . .

Every even integer greater than 2 is the sum of two primes.

???

Limitations of propositional logic

All students know arithmetic.

AliceIsStudent → AliceKnowsArithmetic

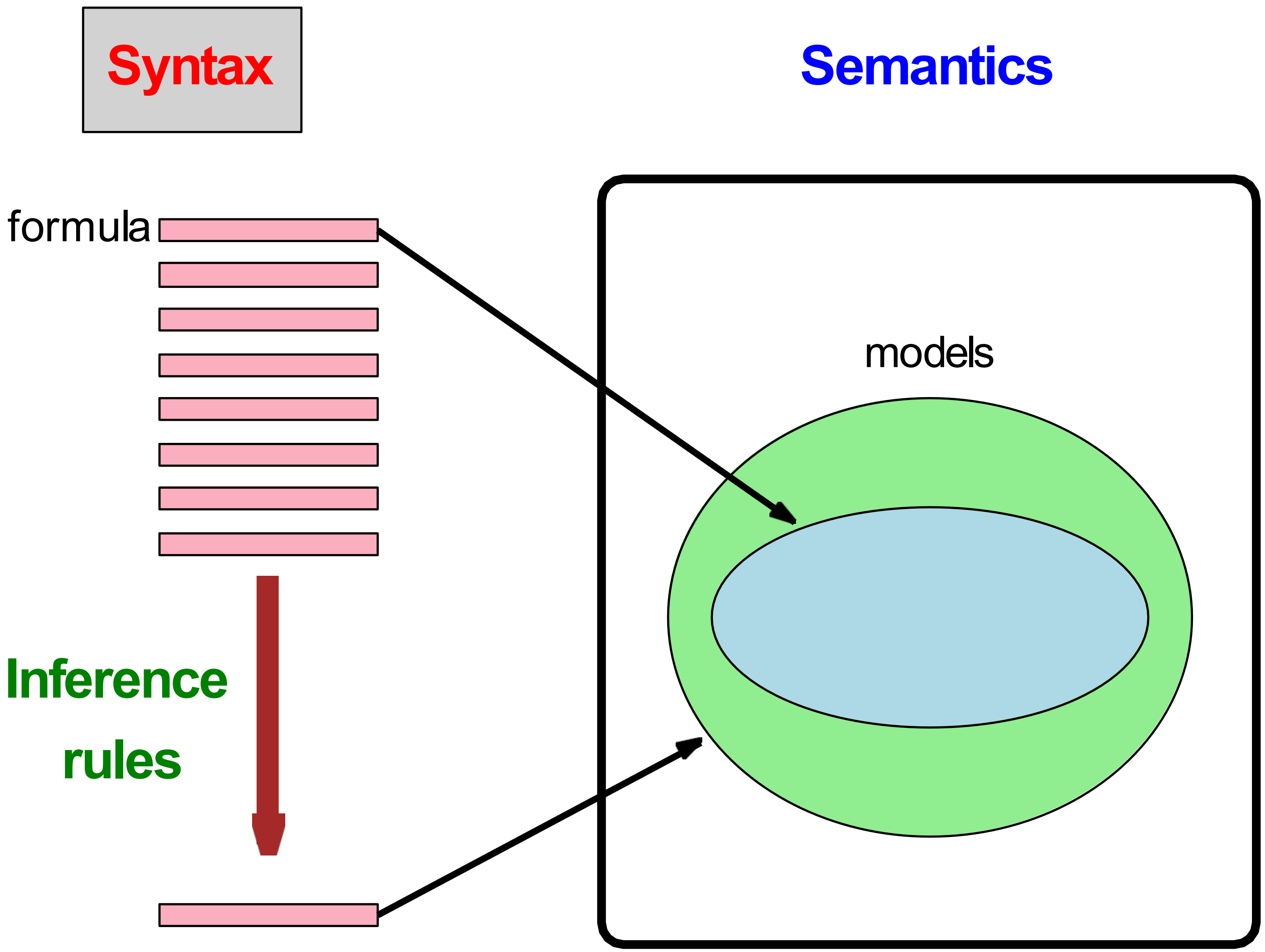
BobIsStudent → BobKnowsArithmetic

. ..

Propositional logic is very clunky. What’s missing?

- Objects and relations: propositions (e.g., AliceKnowsArithmetic) have more internal structure (alice, Knows, arithmetic)
- Quantifiers and variables: *all* is a quantifier which applies to each person, don’t want to enumerate them all...

First-order logic



First-order logic: examples

Alice and Bob both know arithmetic.

$$\text{Knows}(\text{alice}, \text{arithmetic}) \wedge \text{Knows}(\text{bob}, \text{arithmetic})$$

All students know arithmetic.

$$\forall x\, \text{Student}(x) \rightarrow \text{Knows}(x, \text{arithmetic})$$

Syntax of first-order logic

Terms (refer to objects):

- Constant symbol (e.g., arithmetic)
- Variable (e.g., x)
- Function of terms (e.g., Sum(3, x))

Formulas (refer to truth values):

- Atomic formulas (atoms): predicate applied to terms (e.g., Knows(x, arithmetic))
- Connectives applied to formulas (e.g., Student(x) $\rightarrow$ Knows(x, arithmetic))
- Quantifiers applied to formulas (e.g., $\forall x$Student(x) $\rightarrow$ Knows(x, arithmetic))

Quantifiers

Universal quantification ($\forall$):

Think conjunction: $\forall x P(x)$ is like $P(A) \wedge P(B) \wedge \cdots$

Existential quantification ($\exists$):

Think disjunction: $\exists x P(x)$ is like $P(A) \vee P(B) \vee \cdots$

Some properties:

- $\neg \forall x P(x)$ equivalent to $\exists x \neg P(x)$
- $\forall x \exists y$ Knows(x, y) different from $\exists y \forall x$Knows(x, y)