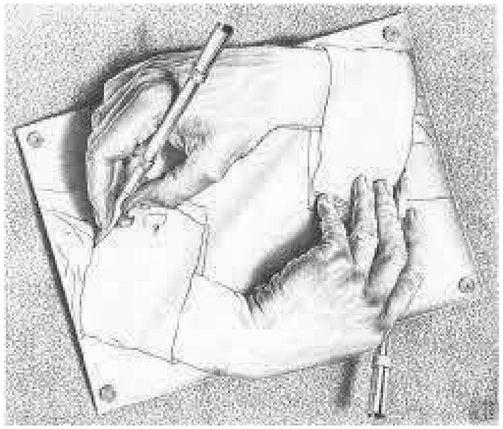
Logic II



Lecture

Review

Propositional modus ponens

Propositional resolution

First order logic

First order modus ponens

First order resolution

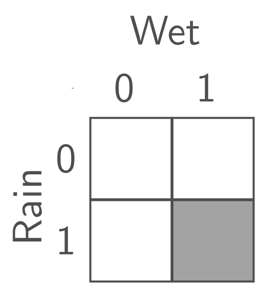
Review: ingredients of a logic

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

Example: Rain $\wedge$ Wet

Semantics: for each formula f, specify a set of **models** $\mathcal{M}(f)$ (assignments / configurations of the world)

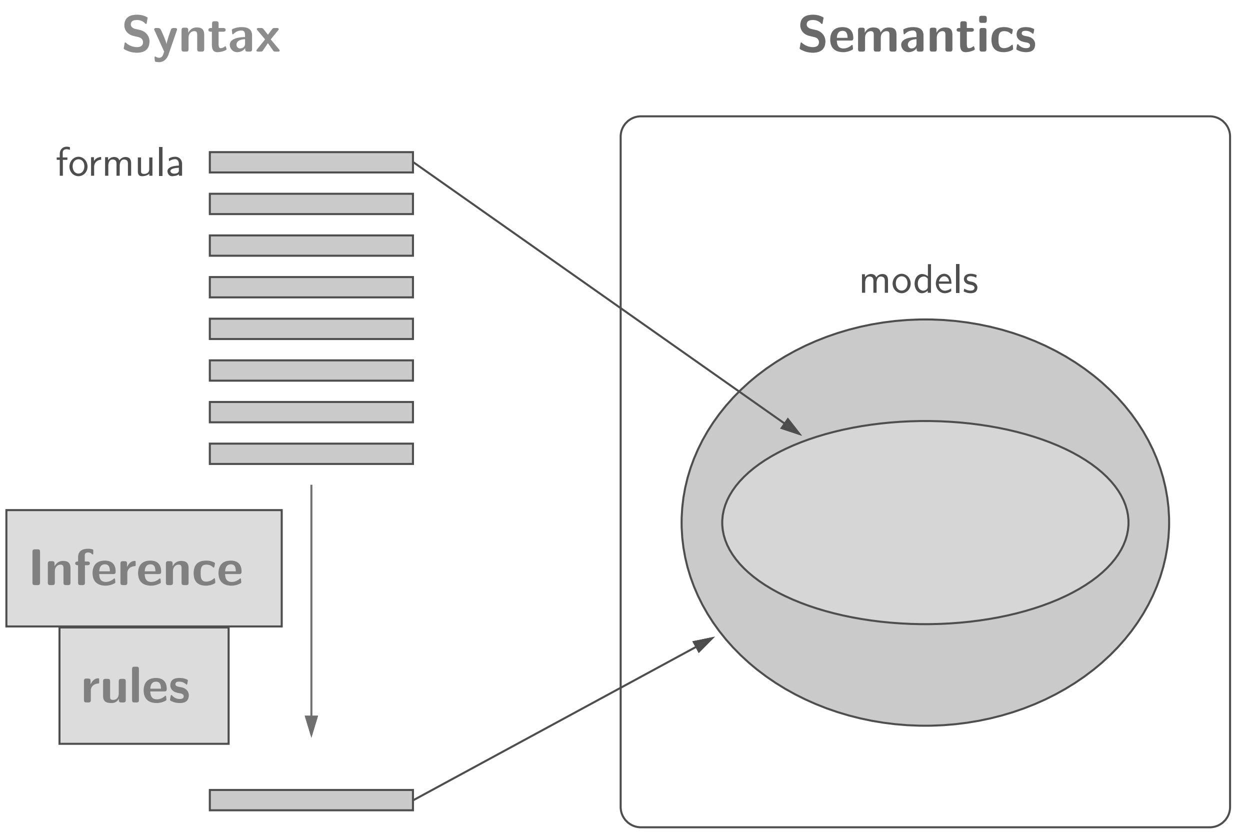
Example:



Inference rules: given KB, what new formulas f can be derived?

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

Propositional logic



Review: formulas

Propositional logic: any legal combination of symbols

$$(\text{Rain} \wedge \text{Snow}) \rightarrow (\text{Traffic} \vee \text{Peaceful}) \wedge \text{Wet}$$

Review: inference algorithm

Inference algorithm:

$$\text{KB} \xrightarrow{\text{(repeatedly apply inference rules)}} f$$

Definition: modus ponens inference rule

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

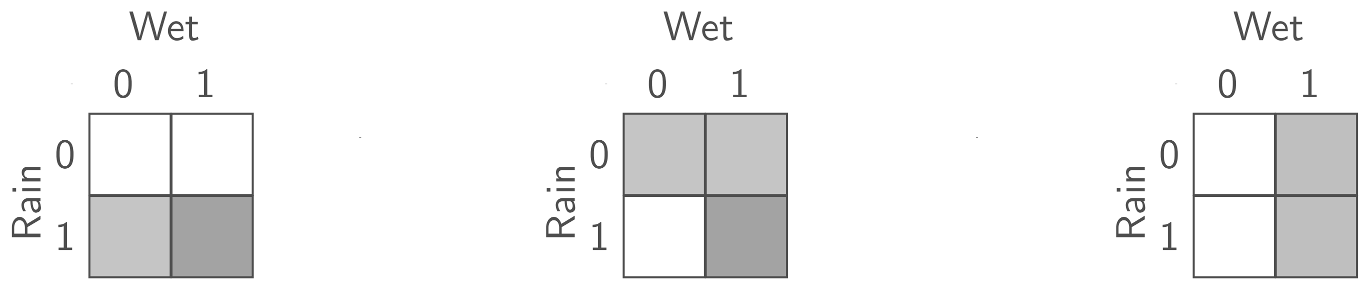
Desiderata: soundness and completeness

entailment (KB $\models f$) derivation (KB $\vdash f$)

Soundness: example

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

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

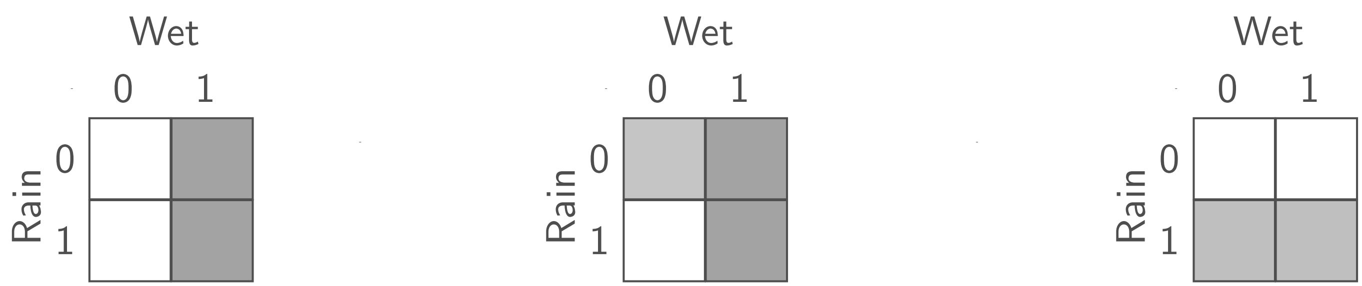


Sound!

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!

Completeness: example

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

Example: Modus ponens is incomplete

Setup:

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

$$f = \text{Wet}$$

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

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

Syntactically: $\text{KB} \not\vdash f$ (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

Lecture

Review

Propositional modus ponens

Propositional resolution

First order logic

First order modus ponens

First order resolution

Definite clauses

Definition: Definite clause

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

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

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

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

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

Example: Traffic

Non-example: $\neg\text{Traffic}$

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

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

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

equivalent: $\neg$(Traffic $\wedge$ 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)!

Example: Modus ponens

KB

Rain

Weekday

Rain $\rightarrow$ Wet

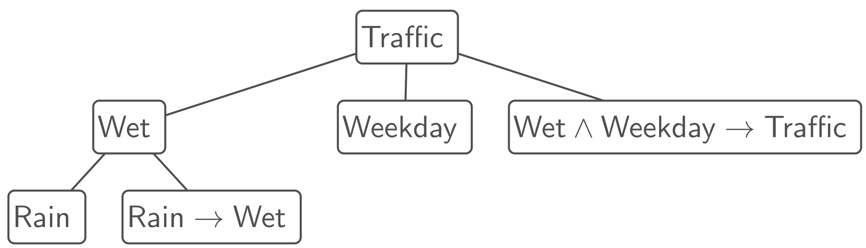
Wet $\wedge$ Weekday $\rightarrow$ Traffic

Traffic $\wedge$ Careless $\rightarrow$ Accident

Definition: Modus ponens

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

Question: KB $\models$ Traffic $\quad \Leftrightarrow \quad$ KB $\vdash$ Traffic



Lecture

Review

Propositional modus ponens

Propositional resolution

First order logic

First order modus ponens

First order resolution

Review: tradeoffs

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

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

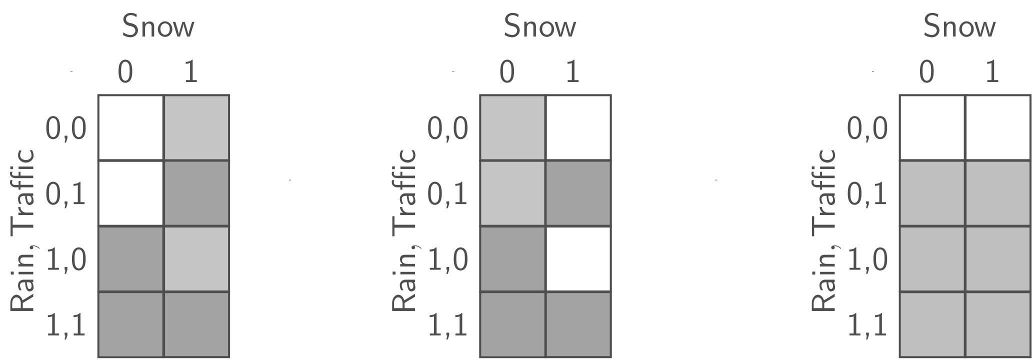
Definition: resolution inference rule

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

Soundness of resolution

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

$\mathcal{M}(\text{Rain} \vee \text{Snow}) \cap \mathcal{M}(\neg\text{Snow} \vee \text{Traffic}) \subseteq ?\ \mathcal{M}(\text{Rain} \vee \text{Traffic})$



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:

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

Remove implication ($\to$):

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

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

Resolution algorithm

Recall: relationship between entailment and contradiction (basically "proof by contradiction")

$$\text{KB} \models f \qquad \longleftrightarrow \qquad \text{KB} \cup \{\neg f\} \text{ is unsatisfiable}$$

Algorithm: resolution-based inference

- Add $\neg f$ into KB.
- Convert all formulas into **CNF**.
- Repeatedly apply **resolution** rule.
- Return entailment iff derive false.

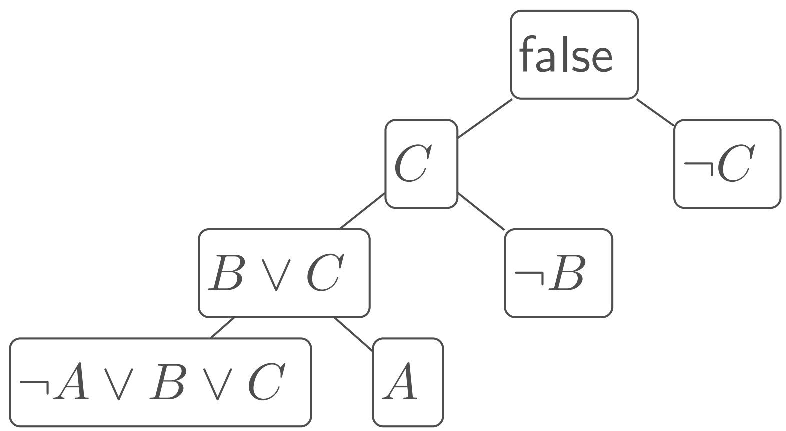
Resolution: example

$$\mathsf{KB}' = \{A \to (B \vee C), A, \neg B, \neg C\}$$

Convert to CNF:

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

Repeatedly apply **resolution** rule:



Conclusion: KB **entails** f

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 $\Rightarrow$ 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 $\Rightarrow$ exponential time

Summary

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

Lecture

Review

Propositional modus ponens

Propositional resolution

First order logic

First order modus ponens

First order resolution

Limitations of propositional logic

Alice and Bob both know arithmetic.

$\text{AliceKnowsArithmetic} \wedge \text{BobKnowsArithmetic}$

All students know arithmetic.

$\text{AliceIsStudent} \rightarrow \text{AliceKnowsArithmetic}$

$\text{BobIsStudent} \rightarrow \text{BobKnowsArithmetic}$

$\dots$

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

???

Limitations of propositional logic

All students know arithmetic.

AliceIsStudent $\rightarrow$ AliceKnowsArithmetic

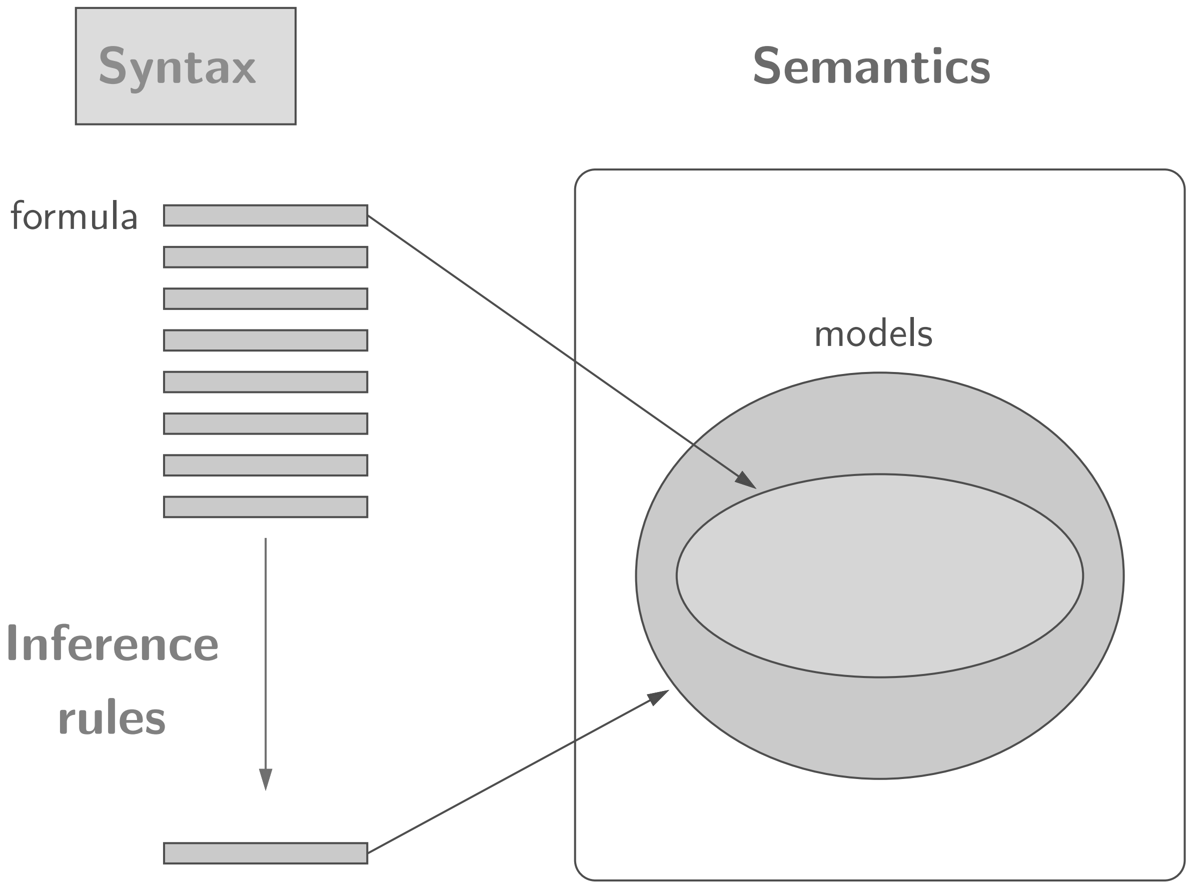
BobIsStudent $\rightarrow$ BobKnowsArithmetic

$\cdots$

Propositional logic is very clunky. What's missing?

- Objects and predicates: 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$(x, \text{arithmetic})$)
- Connectives applied to formulas (e.g., Student$(x) \rightarrow$ Knows$(x, \text{arithmetic})$)
- Quantifiers applied to formulas (e.g., $\forall x$ Student$(x) \rightarrow$ Knows$(x, \text{arithmetic})$)

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 $\exists x\, \neg P(x)$
- $\forall x\, \exists y\, \text{Knows}(x, y)$ different from $\exists y\, \forall x\, \text{Knows}(x, y)$

Natural language quantifiers

Universal quantification ($\forall$):

Every student knows arithmetic.

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

Existential quantification ($\exists$):

Some student knows arithmetic.

$\exists x \, \text{Student}(x) \wedge \text{Knows}(x, \text{arithmetic})$

Note the different connectives!

Some examples of first-order logic

There is some course that every student has taken.

$$\exists y \, \mathsf{Course}(y) \wedge [\forall x \, \mathsf{Student}(x) \rightarrow \mathsf{Takes}(x, y)]$$

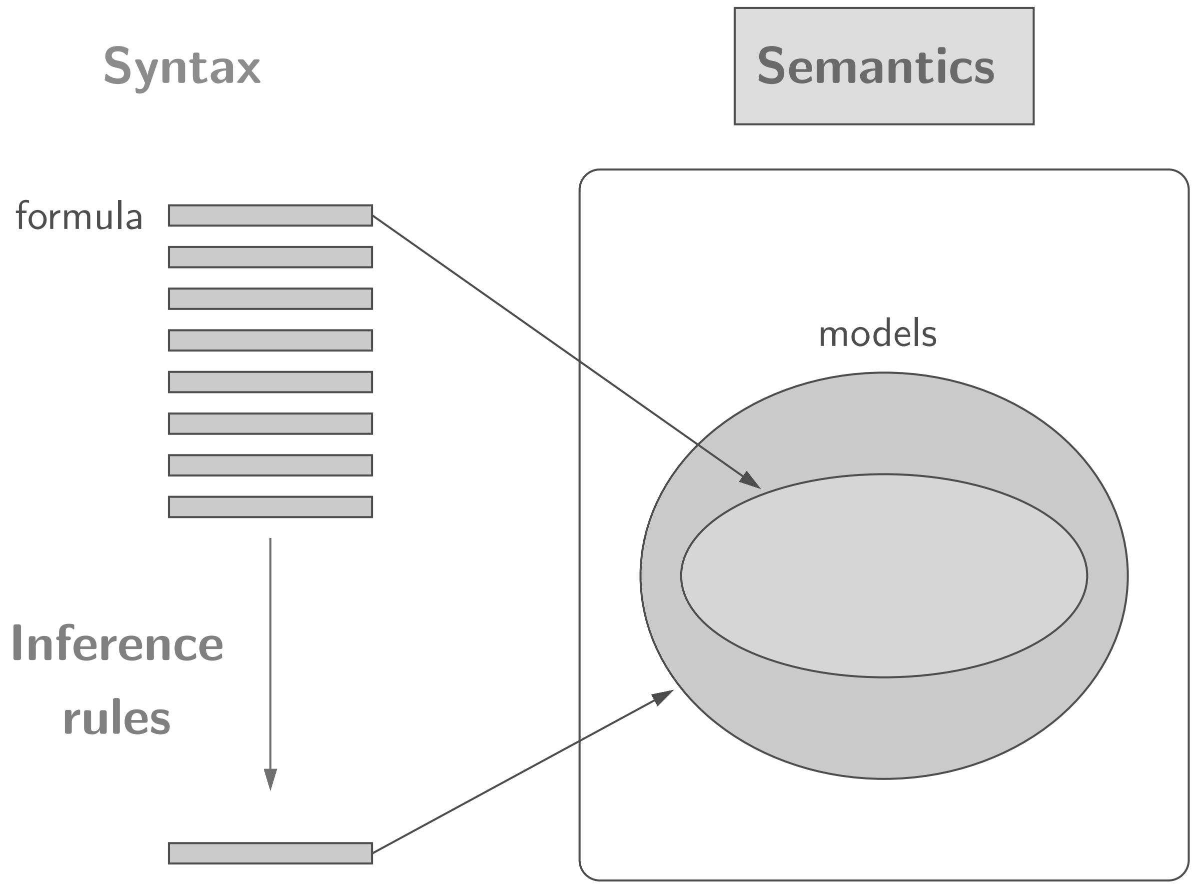
Every even integer greater than 2 is the sum of two primes.

$$\forall x \, \mathsf{EvenInt}(x) \wedge \mathsf{Greater}(x, 2) \rightarrow \exists y \, \exists z \, \mathsf{Equals}(x, \mathsf{Sum}(y, z)) \wedge \mathsf{Prime}(y) \wedge \mathsf{Prime}(z)$$

If a student takes a course and the course covers a concept, then the student knows that concept.

$$\forall x \, \forall y \, \forall z \, (\mathsf{Student}(x) \wedge \mathsf{Takes}(x, y) \wedge \mathsf{Course}(y) \wedge \mathsf{Covers}(y, z)) \rightarrow \mathsf{Knows}(x, z)$$

First-order logic



Models in first-order logic

Recall a model represents a possible situation in the world.

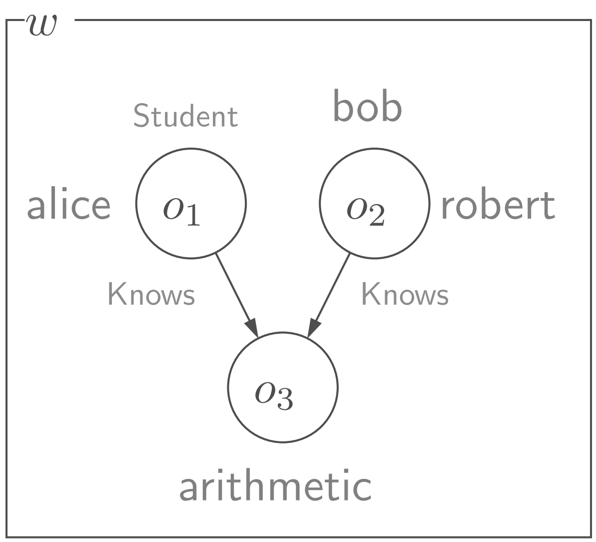
Propositional logic: Model w maps propositional symbols to truth values.

$$w = \{\text{AliceKnowsArithmetic} : 1, \text{BobKnowsArithmetic} : 0\}$$

First-order logic: ?

Graph representation of a model

If only have unary and binary predicates, a model w can be represented as a directed graph:



- Nodes are objects, labeled with constant symbols
- Directed edges are binary predicates, labeled with predicate symbols; unary predicates are additional node labels

Models in first-order logic

Definition: model in first-order logic

A model w in first-order logic maps:

- constant symbols to objects

$w(\text{alice}) = o_1, w(\text{bob}) = o_2, w(\text{arithmetic}) = o_3$

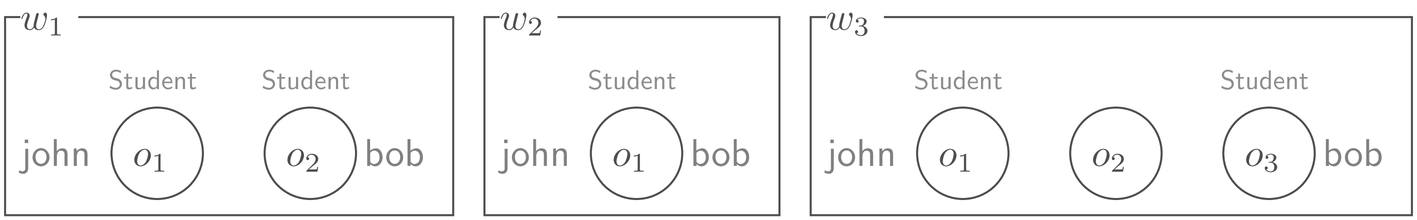
- predicate symbols to tuples of objects

$w(\text{Knows}) = \{(o_1, o_3), (o_2, o_3), \dots\}$

A restriction on models

John and Bob are students.

$$\text{Student}(\text{john}) \wedge \text{Student}(\text{bob})$$



- Unique names assumption: Each object has **at most one** constant symbol. This rules out w_2.
- Domain closure: Each object has **at least one** constant symbol. This rules out w_3.

Point:

$$\text{constant symbol} \longleftrightarrow \text{object}$$

Propositionalization

If one-to-one mapping between constant symbols and objects (**unique names** and **domain closure**),

first-order logic is syntactic sugar for propositional logic:

Knowledge base in first-order logic

$$\text{Student}(\text{alice}) \wedge \text{Student}(\text{bob})$$
$$\forall x\, \text{Student}(x) \rightarrow \text{Person}(x)$$
$$\exists x\, \text{Student}(x) \wedge \text{Creative}(x)$$

Knowledge base in propositional logic

$$\text{Studentalice} \wedge \text{Studentbob}$$
$$(\text{Studentalice} \rightarrow \text{Personalice}) \wedge (\text{Studentbob} \rightarrow \text{Personbob})$$
$$(\text{Studentalice} \wedge \text{Creativealice}) \vee (\text{Studentbob} \wedge \text{Creativebob})$$

Point: use any inference algorithm for propositional logic!

Lecture

Review

Propositional modus ponens

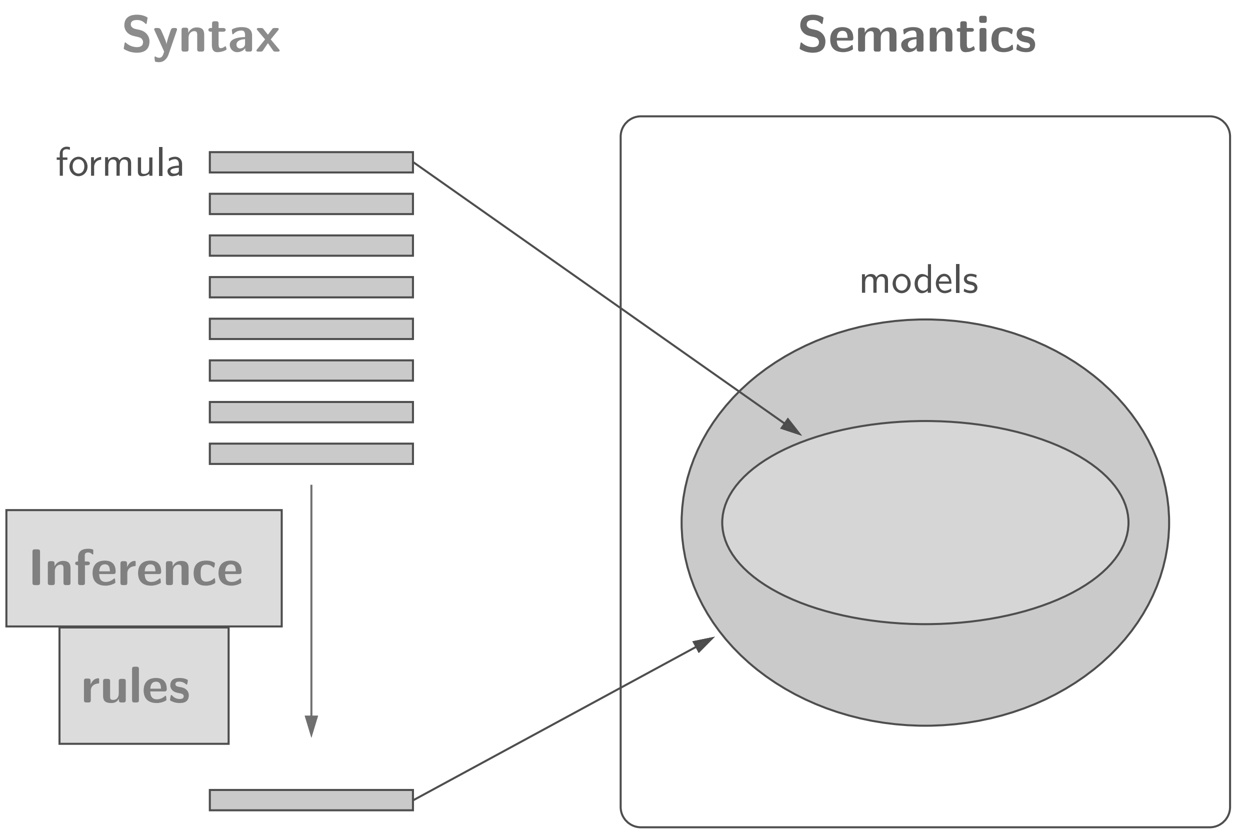
Propositional resolution

First order logic

First order modus ponens

First order resolution

First-order logic



Definite clauses

$$\forall x\, \forall y\, \forall z\, (\mathsf{Takes}(x, y) \wedge \mathsf{Covers}(y, z)) \rightarrow \mathsf{Knows}(x, z)$$

Note: if propositionalize, get one formula for each value to (x, y, z), e.g., $(\mathsf{alice}, \mathsf{cs221}, \mathsf{mdp})$

Definition: definite clause (first-order logic)

A definite clause has the following form:

$$\forall x_1 \cdots \forall x_n\, (a_1 \wedge \cdots \wedge a_k) \rightarrow b$$

for variables $x_1, \dots, x_n$ and atomic formulas $a_1, \dots, a_k, b$ (which contain those variables).

Modus ponens (first attempt)

Definition: modus ponens (first-order logic)

$$\frac{a_1, \dots, a_k \quad \forall x_1 \cdots \forall x_n (a_1 \wedge \cdots \wedge a_k) \rightarrow b}{b}$$

Setup:

Given $P(\text{alice})$ and $\forall x\, P(x) \rightarrow Q(x)$.

Problem:

Can't infer $Q(\text{alice})$ because $P(x)$ and $P(\text{alice})$ don't match!

Solution: substitution and unification

Substitution

Subst$[\{x/\text{alice}\}, P(x)] = P(\text{alice})$

Subst$[\{x/\text{alice}, y/z\}, P(x) \wedge K(x, y)] = P(\text{alice}) \wedge K(\text{alice}, z)$

Definition: Substitution

A substitution θ is a mapping from variables to terms.

Subst$[\theta, f]$ returns the result of performing substitution θ on f.

Unification

Unify[Knows(alice, arithmetic), Knows(x, arithmetic)] = $\{x/$alice$\}$

Unify[Knows(alice, y), Knows(x, z)] = $\{x/$alice$, y/z\}$

Unify[Knows(alice, y), Knows(bob, z)] = fail

Unify[Knows(alice, y), Knows($x, F(x)$)] = $\{x/$alice$, y/F($alice$)\}$

Definition: Unification

Unification takes two formulas f and g and returns a substitution θ which is the most general unifier:

Unify$[f, g] = \theta$ such that Subst$[\theta, f]$ = Subst$[\theta, g]$

or "fail" if no such θ exists.

Modus ponens

Definition: modus ponens (first-order logic)

$$\frac{a'_1, \ldots, a'_k \quad \forall x_1 \cdots \forall x_n (a_1 \wedge \cdots \wedge a_k) \to b}{b'}$$

Get most general unifier θ on premises:

- $\theta = \text{Unify}[a'_1 \wedge \cdots \wedge a'_k, a_1 \wedge \cdots \wedge a_k]$

Apply θ to conclusion:

- $\text{Subst}[\theta, b] = b'$

Modus ponens example

Example: modus ponens in first-order logic

Premises:

$\text{Takes}(\text{alice}, \text{cs221})$

$\text{Covers}(\text{cs221}, \text{mdp})$

$\forall x \, \forall y \, \forall z \, \text{Takes}(x, y) \wedge \text{Covers}(y, z) \rightarrow \text{Knows}(x, z)$

Conclusion:

$\theta = \{x/\text{alice}, y/\text{cs221}, z/\text{mdp}\}$

Derive $\text{Knows}(\text{alice}, \text{mdp})$

Complexity

$$\forall x\, \forall y\, \forall z\, P(x, y, z)$$

- Each application of Modus ponens produces an atomic formula.
- If no function symbols, number of atomic formulas is at most

$$(\text{num-constant-symbols})^{(\text{maximum-predicate-arity})}$$

- If there are function symbols (e.g., F), then infinite...

$$Q(a) \quad Q(F(a)) \quad Q(F(F(a))) \quad Q(F(F(F(a)))) \quad \cdots$$

Complexity

Theorem: completeness

Modus ponens is complete for first-order logic with only Horn clauses.

Theorem: semi-decidability

First-order logic (even restricted to only Horn clauses) is **semi-decidable**.

- If KB $\models f$, forward inference on complete inference rules will prove f in finite time.
- If KB $\not\models f$, no algorithm can show this in finite time.

Lecture

Review

Propositional modus ponens

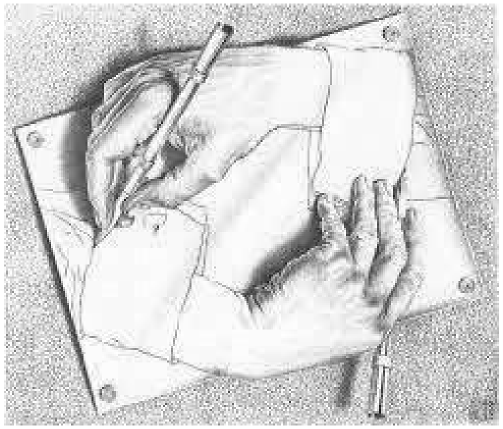
Propositional resolution

First order logic

First order modus ponens

First order resolution

First-order resolution



Resolution

Recall: First-order logic includes non-Horn clauses

$$\forall x \, \mathsf{Student}(x) \rightarrow \exists y \, \mathsf{Knows}(x, y)$$

High-level strategy (same as in propositional logic):

- Convert all formulas to CNF
- Repeatedly apply resolution rule

Conversion to CNF

Input:

$$\forall x\, (\forall y\, \mathsf{Animal}(y) \to \mathsf{Loves}(x, y)) \to \exists y\, \mathsf{Loves}(y, x)$$

Output:

$$(\mathsf{Animal}(Y(x)) \vee \mathsf{Loves}(Z(x), x)) \wedge (\neg \mathsf{Loves}(x, Y(x)) \vee \mathsf{Loves}(Z(x), x))$$

New to first-order logic:

- All variables (e.g., x) have universal quantifiers by default
- Introduce **Skolem functions** (e.g., $Y(x)$) to represent existential quantified variables

Conversion to CNF (part 1)

Anyone who loves all animals is loved by someone.

Input:

$\forall x\, (\forall y\, \text{Animal}(y) \to \text{Loves}(x, y)) \to \exists y\, \text{Loves}(y, x)$

Eliminate implications (old):

$\forall x\, \neg(\forall y\, \neg\text{Animal}(y) \vee \text{Loves}(x, y)) \vee \exists y\, \text{Loves}(y, x)$

Push $\neg$ inwards, eliminate double negation (old):

$\forall x\, (\exists y\, \text{Animal}(y) \wedge \neg\text{Loves}(x, y)) \vee \exists y\, \text{Loves}(y, x)$

Standardize variables (**new**):

$\forall x\, (\exists y\, \text{Animal}(y) \wedge \neg\text{Loves}(x, y)) \vee \exists z\, \text{Loves}(z, x)$

Conversion to CNF (part 2)

$\forall x \, (\exists y \, \text{Animal}(y) \wedge \neg \text{Loves}(x, y)) \vee \exists z \, \text{Loves}(z, x)$

Replace existentially quantified variables with Skolem functions (**new**):

$\forall x \, [\text{Animal}(Y(x)) \wedge \neg \text{Loves}(x, Y(x))] \vee \text{Loves}(Z(x), x)$

Distribute $\vee$ over $\wedge$ (old):

$\forall x \, [\text{Animal}(Y(x)) \vee \text{Loves}(Z(x), x)] \wedge [\neg \text{Loves}(x, Y(x)) \vee \text{Loves}(Z(x), x)]$

Remove universal quantifiers (**new**):

$[\text{Animal}(Y(x)) \vee \text{Loves}(Z(x), x)] \wedge [\neg \text{Loves}(x, Y(x)) \vee \text{Loves}(Z(x), x)]$

Resolution

Definition: resolution rule (first-order logic)

$$\frac{f_1 \vee \cdots \vee f_n \vee p, \quad \neg q \vee g_1 \vee \cdots \vee g_m}{\text{Subst}[\theta, f_1 \vee \cdots \vee f_n \vee g_1 \vee \cdots \vee g_m]}$$

where $\theta = \text{Unify}[p, q]$.

Example: resolution

$$\frac{\text{Animal}(Y(x)) \vee \text{Loves}(Z(x), x), \quad \neg\text{Loves}(u, v) \vee \text{Feeds}(u, v)}{\text{Animal}(Y(x)) \vee \text{Feeds}(Z(x), x)}$$

Substitution: $\theta = \{u/Z(x), v/x\}$.

Summary

Propositional logic		First-order logic
model checking		n/a
	$\Leftarrow$ propositionalization	
modus ponens (Horn clauses)		modus ponens++ (Horn clauses)
resolution (general)		resolution++ (general)

++: unification and substitution

Key idea: variables in first-order logic

Variables yield compact knowledge representations.