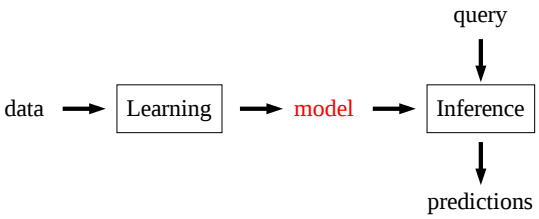
Help

Logic

- Languages and expressiveness
- Propositional logic
  - Specification of propositional logic
  - Inference algorithms for propositional logic
    - Inference in propositional logic with only definite clauses
    - Inference in full propositional logic
- First-order logic
  - Specification of first-order logic
  - Inference algorithms for first-order logic
    - Inference in first-order logic with only definite clauses
    - Inference in full first-order logic
- Other logics
- Logic and probability

Taking a step back



Models describe how the world works (relevant to some task)

What type of models?

Some modeling paradigms

State space models: search problems, MDPs, games

Applications: route finding, game playing, etc.
*Think in terms of **states, actions, and costs***

Variable-based models: CSPs, Markov networks, Bayesian networks

Applications: scheduling, object tracking, medical diagnosis, etc.
*Think in terms of **variables and factors***

Logical models: propositional logic, first-order logic

Applications: proving theorems, program verification, reasoning
*Think in terms of **logical formulas and inference rules***

Outline

- **Languages and expressiveness**
- Propositional logic
  - Specification of propositional logic
  - Inference algorithms for propositional logic
    - Inference in propositional logic with only definite clauses
    - Inference in full propositional logic
- First-order logic
  - Specification of first-order logic
  - Inference algorithms for first-order logic
    - Inference in first-order logic with only definite clauses
    - Inference in full first-order logic
- Other logics
- Logic and probability

Language

***Language** is a mechanism for expressing models/knowledge/ideas.*

Natural languages:

English: *even numbers*
German: *geraden Zahlen*

Programming languages:

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

Desiderata: want a language than can **represent** complex facts about the world and allows for sophisticated **reasoning** about those facts...

Procedural programming languages

Procedural languages represent knowledge using data structures, algorithms manipulates data structures.

```
def f(positions):
  distances = [dist(pos, me) for pos in positions]
  minDistance = min(distances)
  return minDistance
```

Easy query: Given `positions`, what is `minDistance`?

Hard query: Given `minDistance`, what are `positions`?

Need a **declarative** language ($\mathbb{P}(\textbf{positions} \mid \textbf{minDistance})$)

Variable-based models

Variable-based models are a declarative language: define constraints/factors over variables; ask any query over variables.

Implicit representation?

All students work hard.
John is a student.
Therefore, John works hard.

Variable-based models would explictly represent all the students — intuitively shouldn't be necessary.

Higher-order reasoning?

John believes *it will rain*.
Will it rain?
Does John believe *it will not rain* (assuming John is logical)?

Need something more expressive to represent...

Natural languages

A **dime** is better than a **nickel**.

A **nickel** is better than a **penny**.

Therefore, a **dime** is better than **penny**.

General rule (transitivity): if $A > B$ and $B > C$, then $A > C$.

A **penny** is better than **nothing**.

Nothing is better than **world peace**.

Therefore, a **penny** is better than **world peace**???

Natural language is not formal, can be slippery...

A puzzle

If John likes probability, then John likes logic.

If it is Thursday, then John likes probability or John likes logic.

It is Thursday.

Does John like probability?

Does John like logic?

A puzzle

You get extra credit if you write a paper and you solve the problems.

You didn't get extra credit.

You solve the problems.

Did you write a paper?

Propositional logic chat demo

Notes

Tell me some something or ask me something. I will try to convert your utterance into propositional logic and apply resolution to carry out your request. I have no personality.

Example: "If it rained, then the ground is wet.", "It rained.", "Is the ground wet?"
Example: "(implies (or rain snow) wet)", "rain", "(or wet cold)?"

Ingredients

Syntax: defines a set of valid **formulas** ($\mathbf{Formulas}$)

Semantics:

- A **model** w describes a possible situation in the world
- An **interpretation function** $\mathcal{I}$ mapping each $f \in \mathbf{Formulas}$ and model w to a truth value $\mathcal{I}_w(f)$

Inference rules: what new formulas can be added without changing semantics ($\frac{f}{g}$)?

Inference algorithm: apply inference rules in some clever order to answer queries (is f true?)

Syntax versus semantics

Syntax: what are valid expressions in the language?

Semantics: what do these expressions mean?

Different syntax, same semantics:

`x + y` $\Leftrightarrow$ `y + x`

Same syntax, different semantics:

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

Outline

- Languages and expressiveness
- **Propositional logic**
  - Specification of propositional logic
  - Inference algorithms for propositional logic
    - Inference in propositional logic with only definite clauses
    - Inference in full propositional logic
- First-order logic
  - Specification of first-order logic
  - Inference algorithms for first-order logic
    - Inference in first-order logic with only definite clauses
    - Inference in full first-order logic
- Other logics
- Logic and probability

Outline

- Languages and expressiveness
- Propositional logic
  - **Specification of propositional logic**
  - Inference algorithms for propositional logic
    - Inference in propositional logic with only definite clauses
    - Inference in full propositional logic
- First-order logic
  - Specification of first-order logic
  - Inference algorithms for first-order logic
    - Inference in first-order logic with only definite clauses
    - Inference in full first-order logic
- Other logics
- Logic and probability

Syntax of propositional logic

Propositional symbols: A, B, C (think variables in CSPs); these are formulas

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

Note: formulas are just symbols — no meaning yet!

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

Given a formula f and a model w, **interpretation function** $\mathcal{I}_w(f)$ returns either true (1) or false (0).

Base case:

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

Recursive case:

For any two formulas f and g:

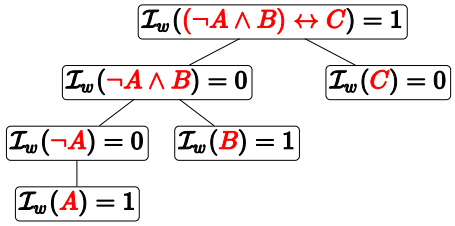
$\mathcal{I}_w(f)$	$\mathcal{I}_w(g)$	$\mathcal{I}_w(\neg f)$	$\mathcal{I}_w(f \wedge g)$	$\mathcal{I}_w(f \vee g)$	$\mathcal{I}_w(f \rightarrow g)$	$\mathcal{I}_w(f \leftrightarrow g)$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Example

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

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

Interpretation:



Formulas represent sets of models

Fach formula f and model w has an interpretation $\mathcal{I}_w(f) \in \{0, 1\}$

Formula:

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

Models with true interpretation $\mathcal{M}(f) = \{w : \mathcal{I}_w(f) = 1\}$:

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

Point: formula is symbols that *compactly* represent a set of models. Think of formula as putting constraints on the world.

Types of formulas

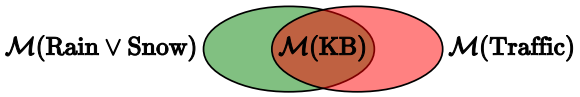
Validity: $\mathcal{I}_w(f) = 1$ for all models w (tautologies that provide no information, e.g., $f = \text{Rain} \vee \neg\text{Rain}$)

Unsatisfiability: $\mathcal{I}_w(f) = 0$ for all models w (contradictions, e.g., $f = \text{Rain} \wedge \neg\text{Rain}$)

Contingent: neither valid nor unsatisfiable (provides information, e.g., $f = \text{Rain}$)

Knowledge base

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



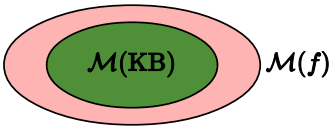
Definition: Knowledge base

A **knowledge base** KB is a set of formulas with

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

Note: think conjunction: $\mathcal{M}(\{f, g\}) = \mathcal{M}(\{f \wedge g\})$

Entailment



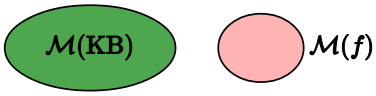
Intuition: f added no information/constraints (it was already known).

Definition: Entailment

$\mathbf{KB}$ entails f (written $\mathbf{KB} \models f$) iff $\mathcal{M}(f) \supset \mathcal{M}(\mathbf{KB})$.

Example: $\mathbf{Rain} \wedge \mathbf{Snow} \models \mathbf{Snow}$

Contradiction



Intuition: f contradicts what we know (captured in $\mathbf{KB}$).

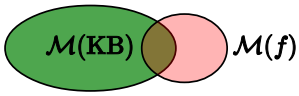
$$\mathcal{M}(\mathbf{KB} \cup \{f\}) = \emptyset$$

Example: $\mathbf{Rain} \wedge \mathbf{Snow}$ contradicts $\neg\mathbf{Snow}$

Relationship between entailment and contradiction:

$\mathbf{KB}$ entails f ($\mathbf{KB} \models f$) iff $\mathbf{KB} \cup \{\neg f\}$ is **unsatisfiable**

Contingency



Intuition: f adds non-trivial information to $\mathbf{KB}$

$$\emptyset \subsetneq \mathcal{M}(\mathbf{KB} \cup \{f\}) \subsetneq \mathcal{M}(\mathbf{KB})$$

Example: $\mathbf{Rain} \vee \mathbf{Snow}$ and $\mathbf{Snow}$

Interacting with a knowledge base

$\mathbf{Tell}[f]$ or $\mathbf{Ask}[f]$ → $\boxed{\mathbf{KB}}$ → response

Tell: *It rained.* ($\mathbf{Tell}[\mathbf{Rain}]$) Possible responses:

- Already knew that: entailment ($\mathbf{KB} \models f$)
- Don't believe that: contradiction ($\mathbf{KB} \models \neg f$)
- Learned something new (update knowledge base): contingent

Ask: *Did it rain?* ($\mathbf{Ask}[\mathbf{Rain}]$) Possible responses:

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

Everything boils down to entailment (**checking satisfiability**)...

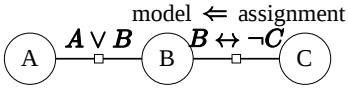
Model checking

Checking satisfiability (SAT) is special case of solving CSPs

propositional symbol ⇒ variable

formula ⇒ constraint

model ⇐ assignment



Solving CSPs is **model checking** (operate over models).

Popular algorithms:

- DPLL (backtracking search)
- WalkSat (Gibbs sampling)

But logic allows us to peer inside factors and exploit structure...

theorem proving operates on formulas.

Outline

- Languages and expressiveness
- Propositional logic
  - Specification of propositional logic
  - **Inference algorithms for propositional logic**
    - Inference in propositional logic with only definite clauses
    - Inference in full propositional logic
- First-order logic
  - Specification of first-order logic
  - Inference algorithms for first-order logic
    - Inference in first-order logic with only definite clauses
    - Inference in full first-order logic
- Other logics
- Logic and probability

Inference rules

Example of making an inference:

It rained. ($\mathbf{Rain}$)
If it rained, then the ground is wet. ($\mathbf{Rain} \rightarrow \mathbf{Wet}$)
Therefore, the ground is wet. ($\mathbf{Wet}$)

$$\frac{\mathbf{Rain}, \quad \mathbf{Rain} \rightarrow \mathbf{Wet}}{\mathbf{Wet}} \quad \begin{matrix}\text{(premises)} \\ \text{(conclusion)}\end{matrix}$$

Modus Ponens inference rule

For any formulas f and g:

$$\frac{f, \quad f \rightarrow g}{g}$$

Key: operate on **syntax**, not **semantics** (can be more efficient).

Inference framework

In general, have a set of rules $\mathbf{Rules}$ with the following form:

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

Forward inference algorithm

Repeat until no changes to $\mathbf{KB}$:
- Choose set of formulas $f_1, \ldots, f_k \in \mathbf{KB}$.
- Find matching rule $\frac{f_1, \quad \ldots \quad, f_k}{g}$.
- Add g to $\mathbf{KB}$.

Say that $\mathbf{KB} \vdash f$ ($\mathbf{KB}$ **derives/proves** f) if there exists sequence of rule applications that eventually adds f to $\mathbf{KB}$.

Soundness and completeness

What properties does a set of inference rules $\mathbf{Rules}$ have?

Definition: Soundness (only prove entailed formulas)

A set of rules $\mathbf{Rules}$ is sound if:

$\mathbf{KB} \vdash f$ implies $\mathbf{KB} \models f$

Definition: Completeness (prove all entailed formulas)

A set of rules $\mathbf{Rules}$ is complete if:

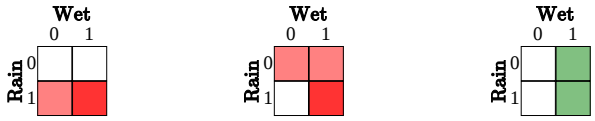
$\mathbf{KB} \models f$ implies $\mathbf{KB} \vdash f$

Point: These properties link syntax (inference rules) and semantics (entailment).

Soundness

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

$$\mathcal{M}(\mathbf{Rain}) \quad \cap \quad \mathcal{M}(\mathbf{Rain} \rightarrow \mathbf{Wet}) \quad \subset? \quad \mathcal{M}(\mathbf{Wet})$$



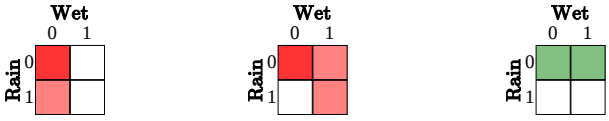
Yes! Models represented by $\mathbf{Rain}$ and $\mathbf{Rain} \rightarrow \mathbf{Wet}$ is **subset** of those represented by $\mathbf{Wet}$.

$$\mathcal{M}(\{\mathbf{Rain}, \mathbf{Rain} \rightarrow \mathbf{Wet}\}) \subset \mathcal{M}(\mathbf{Wet})$$

Soundness

Is $\frac{\neg\mathbf{Wet}, \quad \mathbf{Rain} \rightarrow \mathbf{Wet}}{\neg\mathbf{Rain}}$ (Modus tollens) sound?

$$\mathcal{M}(\neg\mathbf{Wet}) \quad \cap \quad \mathcal{M}(\mathbf{Rain} \rightarrow \mathbf{Wet}) \quad \subset? \quad \mathcal{M}(\neg\mathbf{Rain})$$

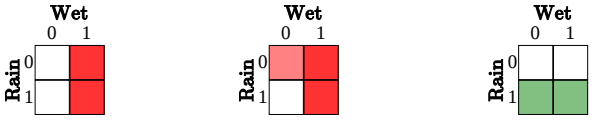


Yes!

Soundness

Is $\frac{\mathbf{Wet}, \quad \mathbf{Rain} \rightarrow \mathbf{Wet}}{\mathbf{Rain}}$ sound?

$$\mathcal{M}(\mathbf{Wet}) \quad \cap \quad \mathcal{M}(\mathbf{Rain} \rightarrow \mathbf{Wet}) \quad \subset? \quad \mathcal{M}(\mathbf{Rain})$$

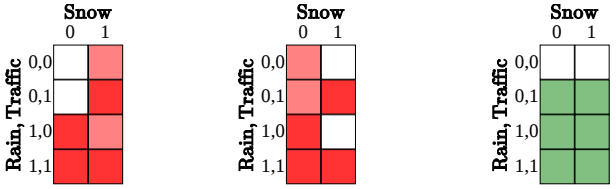


No!

Soundness

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

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



Yes!

Completeness

Example:

$$\text{Rules} = \{\frac{f, \quad f \to g}{g}\}$$

Can verify that all **Rules** is sound, but not complete...for example, given $\text{KB} = \{\text{Rain} \wedge \text{Snow}\}$, can't infer **Rain**.

What set of rules is **complete**? This is tricker than soundness...

Plan:

- Propositional logic with only definite clauses: only need Modus ponens
- Propositional logic: only need resolution rule (+ preprocessing)

Outline

- Languages and expressiveness
- Propositional logic
  - Specification of propositional logic
  - Inference algorithms for propositional logic
    - **Inference in propositional logic with only definite clauses**
    - Inference in full propositional logic
- First-order logic
  - Specification of first-order logic
  - Inference algorithms for first-order logic
    - Inference in first-order logic with only definite clauses
    - Inference in full first-order logic
- Other logics
- Logic and probability

A restriction of propositional logic

Assume knowledge base (KB) contains only definite clauses:

Definition: Definite clause

A definite clause has the following form:

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

for propositional symbols $p_1, \ldots, p_k, q$.

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

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

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

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

Allowed queries to the KB: $\text{Ask}[p]$

Example scenario

Suppose we have the following knowledge base:

Rain
Weekday
$\text{Rain} \to \text{Wet}$
$\text{Wet} \wedge \text{Weekday} \to \text{Traffic}$
$\text{Traffic} \wedge \text{Careless} \to \text{Accident}$

Queries: $\text{Ask}[\text{Traffic}]$? $\text{Ask}[\text{Accident}]$?

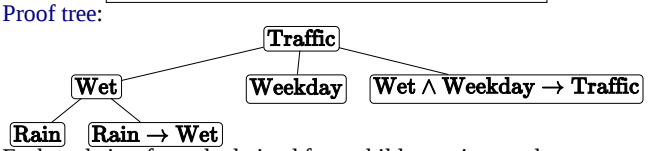
Inference rule

The single rule is sound and complete for propositional logic with only definite clauses:

Definition: Modus ponens

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

Proof tree:



Each node is a formula derived from children using modus ponens, leaves are original formulas in **KB**.

Algorithms

KB: $\{\text{Rain}, \text{Weekday}, \text{Rain} \rightarrow \text{Wet}, \text{Wet} \wedge \text{Weekday} \rightarrow \text{Traffic}\}$
Query: $\text{Ask}[\text{Traffic}]$

Forward chaining:

- From known propositions, iteratively apply rules to derive new propositions.
- Proactively make new inferences when information comes in.
- Time: linear in size of knowledge base.

Backward chaining:

- Start from query and recursively derive premises that conclude the query.
- Make inferences tailored towards answering a particular query.
- Time: often much less than linear in size of knowledge base.

Outline

- Languages and expressiveness
- Propositional logic
  - Specification of propositional logic
  - Inference algorithms for propositional logic
    - Inference in propositional logic with only definite clauses
    - **Inference in full propositional logic**
- First-order logic
  - Specification of first-order logic
  - Inference algorithms for first-order logic
    - Inference in first-order logic with only definite clauses
    - Inference in full first-order logic
- Other logics
- Logic and probability

High-level strategy

Goal: determine whether $\text{KB} \models f$

Example: $\text{KB} = \{A \rightarrow (B \vee C), A, \neg B\}, f = C$

Algorithm (performs proof by contradiction):

- Set $\text{KB}' = \text{KB} \cup \{\neg f\}$.

  Example: $\text{KB}' = \{A \rightarrow (B \vee C), A, \neg B, \neg C\}$
- Run inference algorithm to check **satisfiability** of KB'.
- Conclude $\text{KB} \models f$ iff KB' is unsatisfiable.

  Example: unsatisfiable

Resolution algorithm

Goal: determine whether KB' is satisfiable.

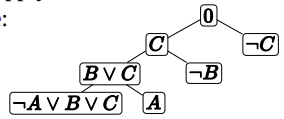
Example: $\text{KB}' = \{A \rightarrow (B \vee C), A, \neg B, \neg C\}$

Algorithm:

- Convert all formulas in KB' into **conjunctive normal form**.

  Example: $\text{KB}' = \{\neg A \vee B \vee C, A, \neg B, \neg C\}$
- Repeatedly apply **resolution** rule.

  Example:



- Return unsatisfiable iff derive false (0).

Conjunctive normal form

Definition: Conjunctive normal form (CNF)

A CNF formula is a conjunction of disjunctions of optional negations of propositional symbols.

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

Conversion to CNF

Goal: convert arbitrary propositional formula into CNF formula

Steps (exercise: verify semantic equivalence):

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

Resolution rule

Resolution rule

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

Example:

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

Summary

- A **model** describes a possible state of the world (e.g., $\{\text{Rain} : 0, \text{Wet} : 1\}$).
- Each **formulas** f (e.g., $\neg\text{Rain}$) describes a set of models $\mathcal{M}(f)$ (think of providing information or imposing constraints).
- **Inference rules** (e.g., $\frac{f, \quad f \rightarrow g}{g}$) allow one to derive new formulas from old ones. Soundness/completeness links syntax and semantics.
- Definite clauses only: ($\text{Rain} \rightarrow \text{Wet}$) forward/backward chaining yields linear time inference.
- Propositional logic: ($\text{Rain} \vee \text{Snow}$) resolution yields exponential time inference.

Outline

- Languages and expressiveness
- Propositional logic
  - Specification of propositional logic
  - Inference algorithms for propositional logic
    - Inference in propositional logic with only definite clauses
    - Inference in full propositional logic
- **First-order logic**
  - Specification of first-order logic
  - Inference algorithms for first-order logic
    - Inference in first-order logic with only definite clauses
    - Inference in full first-order logic
- Other logics
- Logic and probability

Limitations of propositional logic

Alice and Bob both know arithmetic.

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

All students know arithmetic.

$\text{AliceIsStudent} \rightarrow \text{AliceKnowsArithmetic}$
$\text{BobIsStudent} \rightarrow \text{BobKnowsArithmetic}$
$\cdots$

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

???

Limitations of propositional logic

All students know arithmetic.

$\text{AliceIsStudent} \rightarrow \text{AliceKnowsArithmetic}$
$\text{BobIsStudent} \rightarrow \text{BobKnowsArithmetic}$
$\cdots$

Propositional logic is very clunky. What's missing?

- Objects and relations: propositions (e.g., $\text{AliceKnowsArithmetic}$) has more internal structure (Alice, Knows, Arithmetic)
- Quantifiers and variables: *all* is a quantifier which references all people, don't want to enumerate them all...

Outline

- Languages and expressiveness
- Propositional logic
  - Specification of propositional logic
  - Inference algorithms for propositional logic
    - Inference in propositional logic with only definite clauses
    - Inference in full propositional logic
- First-order logic
  - **Specification of first-order logic**
  - Inference algorithms for first-order logic
    - Inference in first-order logic with only definite clauses
    - Inference in full first-order logic
- Other logics
- Logic and probability

Some examples of first-order logic

Alice and Bob both know arithmetic.

$$\textbf{Knows}(\textbf{Alice}, \textbf{Arithmetic}) \wedge \textbf{Knows}(\textbf{Bob}, \textbf{Arithmetic})$$

All students know arithmetic.

$$\forall x\, \textbf{Student}(x) \rightarrow \textbf{Knows}(x, \textbf{Arithmetic})$$

Syntax of first-order logic

Ingredients:

- Connectives from propositional logic: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- Constant symbols (e.g., $\textbf{Alice}, \textbf{Arithmetic}$): refer to objects
- Predicate symbols (e.g., $\textbf{Knows}$): relate multiple objects
- Function symbols (e.g., $\textbf{Sum}$): map objects to single object
- Variables (e.g., x, y, z): refers to objects
- Quantifiers (e.g., $\forall, \exists$): aggregate results from different assignments to variables

Syntax of first-order logic

Terms (refer to objects): constant symbol (e.g., $\textbf{Arithmetic}$), variable (e.g., x), or function applied to terms (e.g., $\textbf{Sum}(3, 4)$)

Formulas (refer to truth values):

- Atomic formulas: Predicate applied to terms (e.g., $\textbf{Knows}(x, \textbf{Arithmetic})$); analogue of propositional symbol in propositional logic
- Connectives applied to formulas (e.g., $\textbf{Student}(x) \rightarrow \textbf{Knows}(x, \textbf{Arithmetic})$); same as propositional logic
- Quantifiers applied to formulas (e.g., $\forall x\, \textbf{Student}(x) \rightarrow \textbf{Knows}(x, \textbf{Arithmetic})$)

Models in first-order logic

Recall a model represents a possible state of affairs (mapping from symbols to their interpretation).

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

e.g., $w(A) = 0, w(B) = 1$

First-order logic:

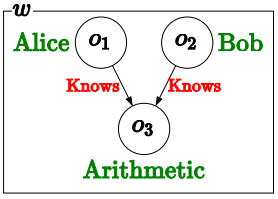
- Model w maps constant symbols to objects

  e.g., $w(\textbf{Alice}) = o_1, w(\textbf{Bob}) = o_2, w(\textbf{Arithmetic}) = o_3$
- Model w maps predicate symbols to tuples of objects

  e.g., $w(\textbf{Knows}) = \{(o_1, o_3), (o_2, o_3), \ldots\}$

Graph representation

A model w as be represented a directed graph (if only have binary predicates):



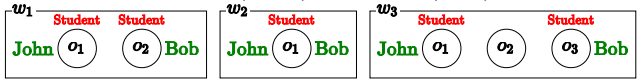
- Nodes are objects, labeled with constant symbols
- Directed edges are relations, labeled with predicate symbols

Database semantics (alternative)

Notes

There are two students, John and Bob.

$$\textbf{Student}(\textbf{John}) \wedge \textbf{Student}(\textbf{Bob})$$



Definition: Unique names assumption

Each object has **at most one** constant symbol. This rules out w_2.

Definition: Domain closure

Each object has **at least one** constant symbol. This rules out w_3.

Definition: Closed-world assumption

All atomic formulas not known (labels not present) are false.

Quantifiers

Universal quantification ($\forall$):

Every student knows arithmetic.

$\forall x\ \mathbf{Student}(x) \rightarrow \mathbf{Knows}(x, \mathbf{Arithmetic})$

Existential quantification ($\exists$):

Some student knows arithmetic.

$\exists x\ \mathbf{Student}(x) \wedge \mathbf{Knows}(x, \mathbf{Arithmetic})$

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

Some examples of first-order logic

Notes

There is some course that every student has taken.

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

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

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

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

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

Outline

- Languages and expressiveness
- Propositional logic
  - Specification of propositional logic
  - Inference algorithms for propositional logic
    - Inference in propositional logic with only definite clauses
    - Inference in full propositional logic
- First-order logic
  - Specification of first-order logic
  - **Inference algorithms for first-order logic**
    - Inference in first-order logic with only definite clauses
    - Inference in full first-order logic
- Other logics
- Logic and probability

Outline

- Languages and expressiveness
- Propositional logic
  - Specification of propositional logic
  - Inference algorithms for propositional logic
    - Inference in propositional logic with only definite clauses
    - Inference in full propositional logic
- First-order logic
  - Specification of first-order logic
  - Inference algorithms for first-order logic
    - **Inference in first-order logic with only definite clauses**
    - Inference in full first-order logic
- Other logics
- Logic and probability

Definite clauses

Assume knowledge base (KB) contains only definite clauses:

Definition: Definite clause

A definite clause has the following form:

$$\forall x_1 \cdots \forall x_n\ (p_1 \wedge \cdots \wedge p_k) \rightarrow q$$

for atomic formulas $p_1, \ldots, p_k, q$ and variables $x_1, \ldots, x_n$ that appear in the atomic formulas.

Example:

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

Intuition: think of first-order definite clause compactly representing all instantiations of the variables (for all objects).

Substitution and unification

Goal: define inference rules that work on formulas with quantifiers

Example:

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

Infer $Q(\text{Alice})$?

Problem: $P(x)$ and $P(\text{Alice})$ don't match exactly.

Two concepts:

- Substitution: morph a formula into another
- Unification: make two formulas the same

Substitution

Definition: Substitution

A substitution θ maps variables to constant symbols or variables.

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

Examples:

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

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

Unification

Definition: Unification

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

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

or fail if no θ exists.

Examples:

$\text{Unify}[\text{Knows}(\text{Alice}, \text{Arithmetic}), \text{Knows}(x, \text{Arithmetic})] = \{x/\text{Alice}\}$

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

$\text{Unify}[\text{Knows}(\text{Alice}, y), \text{Knows}(\text{Bob}, z)] = \text{fail}$

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

Inference rule

Generalized modus ponens

$$\frac{p_1', \quad \cdots \quad , p_k' \quad \forall x_1 \cdots \forall x_n (p_1 \wedge \cdots \wedge p_k) \rightarrow q}{\text{Subst}(\theta, q)},$$

where θ is the most general unifier $\text{Subst}(\theta, p_i) = \text{Subst}(\theta, p_i')$.

Example inputs:

$\text{Takes}(\text{Alice}, \text{CS221})$

$\text{Covers}(\text{CS221}, \text{MDPs})$

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

Example result:

$\theta = \{x/\text{Alice}, y/\text{CS221}, z/\text{MDPs}\}$

Derive $\text{Knows}(\text{Alice}, \text{MDPs})$

Forward/backward chaining

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

Inference algorithms analogous to those for propositional logic.

Forward chaining: starting from known atomic formulas (e.g., $\text{Takes}(\text{Alice}, \text{CS221})$), find rules whose premises unify with them, and derive conclusion.

Backward chaining: starting from query atomic formula (e.g., $\text{Knows}(\text{Alice}, \text{MDPs})$), find rules whose conclusion unifies with it, and recursive on premises.

Time/space complexity

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

- If there are no function symbols, then bounded by number of domain elements to the maximum arity of a predicate (3 in this case).
- If there are function symbols (e.g., F), then infinite...

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

Theorem: Semi-decidability

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

If $\text{KB} \models f$, forward/backward chaining will prove f in finite time.

If $\text{KB} \not\models f$, no algorithm can show this in finite time.

Outline

- Languages and expressiveness
- Propositional logic
  - Specification of propositional logic
  - Inference algorithms for propositional logic
    - Inference in propositional logic with only definite clauses
    - Inference in full propositional logic
- First-order logic
  - Specification of first-order logic
  - Inference algorithms for first-order logic
    - Inference in first-order logic with only definite clauses
    - **Inference in full first-order logic**
- Other logics
- Logic and probability

Resolution

Goal: given a knowledge base, can we derive a contradiction (unsatisfiable)?

Recall: First-order logic can include formulas like this (not a definite clause)

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

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

- Convert all formulas to CNF
- Repeatedly apply resolution rule

Conversion to CNF

Input:

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

Output:

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

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

Input:
$\forall x\, (\forall y\, \text{Animal}(y) \rightarrow \text{Loves}(x, y)) \rightarrow \exists y\, \text{Loves}(y, x)$
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 (old):
$\forall x\, (\exists y\, \text{Animal}(y) \wedge \neg\text{Loves}(x, y)) \vee \exists y\, \text{Loves}(y, x)$
Standardize variables (**new**):
$\forall x\, (\exists y\, \text{Animal}(y) \wedge \neg\text{Loves}(x, y)) \vee \exists z\, \text{Loves}(z, x)$
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**):
$[\text{Animal}(Y(x)) \vee \text{Loves}(Z(x), x)] \wedge [\neg\text{Loves}(x, Y(x)) \vee \text{Loves}(Z(x), x)]$
Interpretation: $Y(x)$ represents animal that x doesn't like, $Z(x)$ represents person who likes x

Resolution

Generalized resolution rule

$$\frac{f_1 \vee \cdots \vee f_n \vee h_1, \quad \neg h_2 \vee g_1 \vee \cdots \vee g_m}{\text{Subst}[\theta, f_1 \vee \cdots \vee f_n \vee g_1 \vee \cdots \vee g_m]}$$

where $\theta = \text{Unify}[h_1, \neg h_2]$.

Example:

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

with substitution $\theta = \{u/Z(x), v/x\}$.

Outline

- Languages and expressiveness
- Propositional logic
  - Specification of propositional logic
  - Inference algorithms for propositional logic
    - Inference in propositional logic with only definite clauses
    - Inference in full propositional logic
- First-order logic
  - Specification of first-order logic
  - Inference algorithms for first-order logic
    - Inference in first-order logic with only definite clauses
    - Inference in full first-order logic
- **Other logics**
- Logic and probability

Motivation

Goal: represent knowledge and perform inferences

Why use anything besides propositional or first-order logic?

Expressiveness:

- Temporal logic: express time
- Modal logic: express alternative worlds
- Higher-order logic: fancier quantifiers

Notational convenience, computational efficiency:

- Description logic

Temporal logic

Barack Obama is the US president.

$\textbf{President}(\textbf{BarackObama}, \textbf{US})$

*George Washington **was** the US president.*

$\mathbf{P}\ \textbf{President}(\textbf{GeorgeWashington}, \textbf{US})$

*Some woman **will be** the US president.*

$\mathbf{F}\ \exists x\ \textbf{Female}(x) \wedge \textbf{President}(x, \textbf{US})$

Temporal logic

Point: all formulas interpreted at a current time.

The following operators change the current time and quantify over it (think of $\mathbf{P}x$ as $\exists t\,(t < \textbf{now}) \wedge f(t)$):

$\mathbf{P}\ f$	f held at some point in the past
$\mathbf{F}\ f$	f will hold at some point in the future
$\mathbf{H}\ f$	f held at every point in the past
$\mathbf{G}\ f$	f will hold at every point in the future

Every student will at some point never be a student again.

$\forall x.\ \textbf{Student}(x) \rightarrow \mathbf{FG}\neg\textbf{Student}(x)$

Model: map from time points to models in first-order logic

Modal logic for propositional attitudes

Alice believes one plus one is two.

$\textbf{Knows}(\textbf{Alice}, \textbf{Equals}(\textbf{Sum}(1,1),2))$??

Alice believes Boston is a city.

$\textbf{Knows}(\textbf{Alice}, \textbf{City}(\textbf{Boston}))$??

Problem: $\textbf{Equals}(\textbf{Sum}(1,1),2)$ is true, $\textbf{City}(\textbf{Boston})$ is true, but two are not interchangeable in this context.

Solution: every formula interpreted with respect to a possible world, operator $\mathbf{K}_{\textbf{Alice}}\ f$ interprets f according to Alice's world

$\mathbf{K}_{\textbf{Alice}}\ \textbf{Equals}(\textbf{Sum}(1,1),2))$

$\mathbf{K}_{\textbf{Alice}}\ \textbf{City}(\textbf{Boston})$

Model: map from possible worlds to models in first-order logic

Higher-order logic: lambda calculus

Simple:

Alice has visited some museum.

$\exists x\ \textbf{Museum}(x) \wedge \textbf{Visited}(\textbf{Alice}, x)$

More complex:

*Alice has visited **at least 10** museums.*

$\lambda x\ \textbf{Museum}(x) \wedge \textbf{Visited}(\textbf{Alice}, x)$: boolean function representing set of museums Alice has visited

$\textbf{Count}(\lambda x\ \textbf{Museum}(x)\textbf{Visited}(\textbf{Alice}, x)) \geq 10$

Higher-order logic allows us to model these generalized quantifiers.

Description logic

People with at least three sons who are all unemployed and married to doctors, and at most two daughters who are professors...are weird.

Lambda calculus:

$\forall x\ (\textbf{Person}(x)$
$\wedge \textbf{Count}(\lambda y\ \textbf{Son}(x,y) \wedge \textbf{Unemployed}(y) \wedge \forall z\ \textbf{Spouse}(y,z) \wedge \textbf{Doctor}(z)) \geq 3$
$\wedge \textbf{Count}(\lambda y\ \textbf{Daughter}(x,y) \wedge \textbf{Professor}(y)) \leq 2) \rightarrow \textbf{Weird}(x)$

Description logic:

$(\textbf{Person}$
$\sqcap (\geq 3\ \textbf{Son}.\ \textbf{Unemployed} \sqcap \forall \textbf{Spouse}.\ \textbf{Doctor})$
$\sqcap (\leq 2\ \textbf{Daughter}.\ \textbf{Professor})) \sqsubseteq \textbf{Weird}$

Advantages:

- Generalized quantifiers without variables: notationally more compact
- First-order is semi-decidable, description logic is decidable

Summary of logics

- Propositional logic: $A \wedge B$
- First-order logic: $\forall x\, P(x) \rightarrow Q(x)$
- Temporal / modal logic: $\mathbf{F}(A \wedge B)$
- Description logic: $P \sqsubseteq Q$
- Higher-order logic (lambda calculus): $\lambda x\, P(x) \wedge Q(x)$

Outline

- Languages and expressiveness
- Propositional logic
  - Specification of propositional logic
  - Inference algorithms for propositional logic
    - Inference in propositional logic with only definite clauses
    - Inference in full propositional logic
- First-order logic
  - Specification of first-order logic
  - Inference algorithms for first-order logic
    - Inference in first-order logic with only definite clauses
    - Inference in full first-order logic
- Other logics
- **Logic and probability**

Limitations

In logic, every formula is true or false. In reality, there is uncertainty.

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

Probability used to define joint distributions: $\mathbb{P}_\theta(X_1, \ldots, X_n)$

Think of $X_1, \ldots, X_n$ as propositional symbols

A model is $w = \{X_1 : x_1, \ldots, X_n : x_n\}$

We are placing a **distribution over possible worlds** w.

Probability theory:

- Pro: allows us to manage uncertainty in a coherent way
- Con: captures propositional logic

Markov logic

Assume database semantics. Defines Markov network:

Random variables:

$W = (P(A), P(B), R(A, A), R(A, B), R(B, A), R(B, B), Q(A), Q(B))$

First-order formula:

$f = [\forall x \forall y\, P(x) \wedge R(x, y) \rightarrow P(y)]$

$g = [\forall x\, Q(x)]$

Equivalent propositional logic formulas:

$f_1 = [P(A) \wedge R(A, A) \rightarrow P(A)]$

$f_2 = [P(A) \wedge R(A, B) \rightarrow P(B)]$

$f_3 = [P(B) \wedge R(B, A) \rightarrow P(A)]$

$f_4 = [P(B) \wedge R(B, B) \rightarrow P(B)]$

$g_1 = Q(A)$

$g_2 = Q(B)$

Markov logic

One parameter for each first-order formula (e.g., f, g)

$\mathcal{I}_w(f) \in \{0, 1\}$ is the interpretation of f in w

Markov logic defines a **Markov network**:

$$\mathbb{P}_\theta(W = w) \propto \exp\{\theta_f \sum_i \mathcal{I}_w(f_i) + \theta_g \sum_i \mathcal{I}_w(g_i)\}$$

- Defines distribution over possible worlds (models)
- All grounded instances of a formula have same parameter weight
- Can do lifted probabilistic inference for efficiency (important for learning)
- As $\theta_f, \theta_g \rightarrow \infty$, get ordinary logic

Summary

- Logic is a language for expressing facts in a knowledge base
- Considerations: expressiveness, notational convenience, inferential complexity
- Propositional logic with definite clauses, propostional logic, description logic, first-order logic, temporal logic, modal logic, higher-order logic
- Markov logic: marry logic (abstract reasoning by working on formulas) and probability (maintaining uncertainty)