Announcements

Assignments:
- HW5
  - Due Tue 2/26, 10 pm
- HW6 and P3
  - Coming soon

Travel
- Pat out Wed 2/27, back for Mon 3/4
  - SIGCSE 2019, Minneapolis
AI: Representation and Problem Solving

First-Order Logic

Instructors: Pat Virtue & Stephanie Rosenthal

Slide credits: CMU AI, http://aima.eecs.berkeley.edu
Outline

1. Need for first-order logic
2. Syntax and semantics
3. Planning with FOL
4. Inference with FOL
Pros and Cons of Propositional Logic

- Propositional logic is **declarative**: pieces of syntax correspond to facts
- Propositional logic allows partial/disjunctive/negated information (unlike most data structures and databases)
- Propositional logic is **compositional**: meaning of $B_{1,1} \land P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$
- Meaning in propositional logic is **context-independent** (unlike natural language, where meaning depends on context)
- Propositional logic has very limited expressive power (unlike natural language)
  
  E.g., cannot say “pits cause breezes in adjacent squares” except by writing one sentence for each square
Pros and Cons of Propositional Logic

Rules of chess:
- 100,000 pages in propositional logic
- 1 page in first-order logic

Rules of pacman:
\[ \forall x,y,t \ \text{At}(x,y,t) \iff [\text{At}(x,y,t-1) \land \neg \exists u,v \ \text{Reachable}(x,y,u,v,\text{Action}(t-1))] \lor [\exists u,v \ \text{At}(u,v,t-1) \land \text{Reachable}(x,y,u,v,\text{Action}(t-1))] \]
First-Order Logic (First-Order Predicate Calculus)

Whereas propositional logic assumes world contains **facts**, first-order logic (like natural language) assumes the world contains

- **Objects:** people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries, ...
- **Relations:** red, round, bogus, prime, multistoried ..., brother of, bigger than, inside, part of, has color, occurred after, owns, ...
- **Functions:** father of, best friend, third inning of, one more than, end of, ...
# Logics in General

<table>
<thead>
<tr>
<th>Language</th>
<th>What exists in the world</th>
<th>What an agent believes about facts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propositional logic</td>
<td>Facts</td>
<td>true / false / unknown</td>
</tr>
<tr>
<td>First-order logic</td>
<td>facts, objects, relations</td>
<td>true / false / unknown</td>
</tr>
<tr>
<td>Probability theory</td>
<td>facts</td>
<td>degree of belief</td>
</tr>
<tr>
<td>Fuzzy logic</td>
<td>facts + degree of truth</td>
<td>known interval value</td>
</tr>
</tbody>
</table>
Syntax of FOL

Basic Elements

- **Constants**: KingJohn, 2, CMU, ...
- **Predicates**: Brother, >, ...
- **Functions**: Sqrt, LeftLegOf, ...
- **Variables**: x, y, a, b, ...
- **Connectives**: ∧, ∨, ¬, ⇒, ⇔
- **Equality**: =
- **Quantifiers**: ∀, ∃
Syntax of FOL

Atomic sentence = \( \text{predicate}(\text{term}_1, \ldots, \text{term}_n) \)
or \( \text{term}_1 = \text{term}_2 \)

\[
\text{Term} = \text{function}(\text{term}_1, \ldots, \text{term}_n)
\text{or constant}
\text{or variable}
\]

Examples

\( \text{Brother}(\text{KingJohn}, \text{RichardTheLionheart}) \)

\( > \ (\text{Length}(\text{LeftLegOf}(\text{Richard})), \ \text{Length}(\text{LeftLegOf}(\text{KingJohn}))) \)
Syntax of FOL

Complex sentences are made from atomic sentences using connectives

\[ \neg S, \quad S_1 \land S_2, \quad S_1 \lor S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2 \]

Examples

\[ \text{Sibling}(\text{King John}, \text{Richard}) \Rightarrow \text{Sibling}(\text{Richard}, \text{King John}) \]

\[ > (1, 2) \lor \leq (1, 2) \]

\[ > (1, 2) \land \neg > (1, 2) \]
Models for FOL

Example
Models for FOL

\textit{Brother(Richard, John)}

Consider the interpretation in which:

\textit{Richard} $\rightarrow$ Richard the Lionheart
\textit{John} $\rightarrow$ the evil King John
\textit{Brother} $\rightarrow$ the brotherhood relation
Model for FOL

Lots of models!
Model for FOL

Lots of models!

Entailment in propositional logic can be computed by enumerating models.

We can enumerate the FOL models for a given KB vocabulary:

For each number of domain elements $n$ from 1 to $\infty$

   For each $k$-ary predicate $P_k$ in the vocabulary
      For each possible $k$-ary relation on $n$ objects
         For each constant symbol $C$ in the vocabulary
            For each choice of referent for $C$ from $n$ objects . . .

Computing entailment by enumerating FOL models is not easy!
Truth in First-Order Logic

Sentences are true with respect to a model and an interpretation.

Model contains \( \geq 1 \) objects (domain elements) and relations among them.

Interpretation specifies referents for:
- constant symbols \( \rightarrow \) objects
- predicate symbols \( \rightarrow \) relations
- function symbols \( \rightarrow \) functional relations

An atomic sentence \( \text{predicate}(\text{term}_1, \ldots, \text{term}_n) \) is true:
- iff the objects referred to by \( \text{term}_1, \ldots, \text{term}_n \)
- are in the relation referred to by \( \text{predicate} \).
Consider the interpretation in which:

\[ \begin{align*}
\text{Richard} & \rightarrow \text{Richard the Lionheart} \\
\text{John} & \rightarrow \text{the evil King John} \\
\text{Brother} & \rightarrow \text{the brotherhood relation}
\end{align*} \]

Under this interpretation, \( \text{Brother}(\text{Richard}, \text{John}) \) is true just in the case Richard the Lionheart and the evil King John are in the brotherhood relation in the model.
Universal Quantification

\( \forall (\text{variables}) \ (\text{sentence}) \)

Everyone at the banquet is hungry:
\[ \forall x \quad \text{At}(x, \text{Banquet}) \Rightarrow \text{Hungry}(x) \]

\( \forall x \quad P \) is true in a model \( m \) iff \( P \) is true with \( x \) being each possible object in the model

Roughly speaking, equivalent to the conjunction of instantiations of \( P \)
\[
(\text{At}(\text{KingJohn, Banquet}) \Rightarrow \text{Hungry}(\text{KingJohn})) \\
\land (\text{At}(\text{Richard, Banquet}) \Rightarrow \text{Hungry}(\text{Richard})) \\
\land (\text{At}(\text{Banquet, Banquet}) \Rightarrow \text{Hungry}(\text{Banquet})) \\
\land \ldots
\]
Universal Quantification

Common mistake

Typically, \( \Rightarrow \) is the main connective with \( \forall \)

Common mistake: using \( \land \) as the main connective with \( \forall \):

\[ \forall x \ At(x, \ Banquet) \land Hungry(x) \]

means “Everyone is at the banquet and everyone is hungry”
Existential Quantification

\[ \exists \text{(variables)} \quad (\text{sentence}) \]

Someone at the tournament is hungry:

\[ \exists x \text{At}(x, \text{Tournament}) \land \text{Hungry}(x) \]

\[ \exists x P \text{ is true in a model } m \iff P \text{ is true with } x \text{ being some possible object in the model} \]

Roughly speaking, equivalent to the disjunction of instantiations of \( P \)

\[ \text{At}(\text{KingJohn}, \text{Tournament}) \land \text{Hungry}(\text{KingJohn}) \]
\[ \lor \text{At}(\text{Richard}, \text{Tournament}) \land \text{Hungry}(\text{Richard}) \]
\[ \lor \text{At}(\text{Tournament}, \text{Tournament}) \land \text{Hungry}(\text{Tournament}) \]
\[ \lor \ldots \]
Existential Quantification

Common mistake

Typically, \( \land \) is the main connective with \( \exists \)

Common mistake: using \( \Rightarrow \) as the main connective with \( \exists \):

\[ \exists x \text{At}(x, Tournament) \Rightarrow \text{Hungry}(x) \]

is true if there is anyone who is not at the tournament!
Properties of Quantifiers

∀x  ∀y is the same as  ∀y  ∀x
∃x  ∃y is the same as  ∃y  ∃x
∃x  ∀y is not the same as  ∀y  ∃x

∃x  ∀y Loves(x, y)
“There is a person who loves everyone in the world”

∀y  ∃x Loves(x, y)
“Everyone in the world is loved by at least one person”

Quantifier duality: each can be expressed using the other

∀x Likes(x, IceCream)  ¬∃x ¬Likes(x, IceCream)
∃x Likes(x, Broccoli)  ¬∀x ¬Likes(x, Broccoli)
Fun with Sentences

Brothers are siblings

∀x, y. Brother(x, y) ⇒ Sibling(x, y).

“Sibling” is symmetric

∀x, y. Sibling(x, y) ⇔ Sibling(y, x).

A first cousin is a child of a parent’s sibling

∀x, y. FirstCousin(x, y) ⇔ ∃p, ps. Parent(p, x) ∧ Sibling(ps, p) ∧ Parent(ps, y).
Equality

\( \text{term}_1 = \text{term}_2 \) is true under a given interpretation if and only if \( \text{term}_1 \) and \( \text{term}_2 \) refer to the same object.

E.g., \( 1 = 2 \) and \( \forall x \times (\text{Sqrt}(x), \text{Sqrt}(x)) = x \) are satisfiable.

\( 2 = 2 \) is valid.

E.g., definition of (full) \textit{Sibling} in terms of \textit{Parent}:

\[
\forall x, y \quad \text{Sibling}(x, y) \iff \\
[\neg(x = y) \land \exists m, f \neg(m = f) \land \\
\text{Parent}(m, x) \land \text{Parent}(f, x) \land \text{Parent}(m, y) \land \text{Parent}(f, y)]
\]
Given the following two FOL sentences:

\[ \gamma: \ \forall x \ \text{Hungry}(x) \]
\[ \delta: \ \exists x \ \text{Hungry}(x) \]

Which of these is true?

A) \( \gamma \models \delta \)
B) \( \delta \models \gamma \)
C) Both
D) Neither
Given the following two FOL sentences:

\[ \gamma: \ \forall x \ Hungry(x) \]
\[ \delta: \ \exists x \ Hungry(x) \]

Which of these is true?

A) \( \gamma \models \delta \)
B) \( \delta \models \gamma \)
C) Both
D) Neither
Interacting with FOL KBs

Suppose a wumpus-world agent is using an FOL KB and perceives a smell and a breeze (but no glitter) at \( t = 5 \):

\[ \text{Tell}(\text{KB}, \text{Percept}([\text{Smell}, \text{Breeze}, \text{None}], 5)) \]

\[ \text{Ask}(\text{KB}, \exists \text{ a Action}(\text{a}, 5)) \]

i.e., does \( \text{KB} \) entail any particular actions at \( t = 5 \)?

Answer: Yes, \( \{\text{a}/\text{Shoot}\} \) ← substitution (binding list)  

Given a sentence \( S \) and a substitution \( \sigma \),

\( S\sigma \) denotes the result of plugging \( \sigma \) into \( S \); e.g.,

\( S = \text{Smarter}(x, y) \)

\( \sigma = \{x/\text{EVE}, y/\text{WALL-E}\} \)

\( S\sigma = \text{Smarter}(\text{EVE}, \text{WALL-E}) \)

\( \text{Ask}(\text{KB}, S) \) returns some/all \( \sigma \) such that \( \text{KB} \models S\sigma \)
Knowledge Base for Wumpus World

“Perception”
\[
\forall b, g, t \quad \text{Percept}([\text{Smell}, b, g], t) \implies \text{Smelt}(t)
\]
\[
\forall s, b, t \quad \text{Percept}([s, b, \text{Glitter}], t) \implies \text{AtGold}(t)
\]

Reflex: \[
\forall t \quad \text{AtGold}(t) \implies \text{Action}(\text{Grab}, t)
\]

Reflex with internal state: do we have the gold already?
\[
\forall t \quad \text{AtGold}(t) \land \neg \text{Holding}(\text{Gold}, t) \implies \text{Action}(\text{Grab}, t)
\]

\text{Holding}(\text{Gold}, t) \text{ cannot be observed} \implies \text{keeping track of change is essential}
Deducing Hidden Properties

Properties of locations:
\( \forall x, t \quad At(Agent, x, t) \land Smelt(t) \Rightarrow Smelly(x) \)
\( \forall x, t \quad At(Agent, x, t) \land Breeze(t) \Rightarrow Breezy(x) \)

Squares are breezy near a pit:

**Diagnostic rule**—infer cause from effect
\( \forall y \quad Breezy(y) \Rightarrow \exists x \quad Pit(x) \land Adjacent(x, y) \)

**Causal rule**—infer effect from cause
\( \forall x, y \quad Pit(x) \land Adjacent(x, y) \Rightarrow Breezy(y) \)

Neither of these is complete — e.g., the causal rule doesn’t say whether squares far away from pits can be breezy

**Definition for the Breezy predicate:**
\( \forall y \quad Breezy(y) \iff [\exists x \quad Pit(x) \land Adjacent(x, y)] \)
Keeping Track of Change

Facts hold in situations, rather than eternally
E.g., Holding(Gold, Now) rather than just Holding(Gold)

**Situation calculus** is one way to represent change in FOL:

- Adds a situation argument to each non-eternal predicate  
  E.g., *Now* in  
  Holding(Gold, Now) denotes a situation

Situations are connected by the *Result* function
*Result*(a, s) is the situation that results from doing a in s
Describing Actions

“Effect” axiom—describe changes due to action
\[ \forall s \; AtGold(s) \Rightarrow \text{Holding}(Gold, \text{Result}(Grab, s)) \]

“Frame” axiom—describe non-changes due to action
\[ \forall s \; \text{HaveArrow}(s) \Rightarrow \text{HaveArrow}(\text{Result}(\text{Grab}, s)) \]

**Successor-state axioms** solve the representational frame problem

Each axiom is “about” a *predicate* (not an action per se):

\[ P \text{ true afterwards } \iff [\text{an action made } P \text{ true} \lor P \text{ true already and no action made } P \text{ false}] \]

For holding the gold:
\[ \forall a, s \; \text{Holding}(Gold, \text{Result}(a, s)) \iff \]
\[ [(a = \text{Grab} \land \text{AtGold}(s)) \lor (\text{Holding}(Gold, s) \land \neg (a = \text{Release}))] \]
Describing Actions

Initial condition in KB:
- \( At(Agent, [1, 1], S_0) \)
- \( At(Gold, [1, 2], S_0) \)

Query: \( Ask(KB, \exists s \ Holding(Gold, s)) \)

i.e., in what situation will I be holding the gold?

Answer: \( \{s/Result(Grab, Result(Forward, S_0))\} \)

i.e., go forward and then grab the gold

This assumes that the agent is interested in plans starting at \( S_0 \) and that \( S_0 \) is the only situation described in the KB.
Making Plans

Represent plans as action sequences \([a_1, a_2, \ldots, a_n]\) 

PlanResult\((p, s)\) is the result of executing \(p\) in \(s\)

Then the query \(\text{Ask}(KB, \exists p \ \text{Holding}(\text{Gold}, \text{PlanResult}(p, S_0)))\) has the solution \(\{p/\{\text{Forward, Grab}\}\}\)

Definition of PlanResult in terms of Result:

\[\forall s \ \text{PlanResult}([\ ], s) = s\]
\[\forall a, p, s \ \text{PlanResult}([a, p], s) = \text{PlanResult}(p, \text{Result}(a, s))\]
Outline

1. Need for first-order logic
2. Syntax and semantics
3. Planning with FOL
4. Inference with FOL
Inference in First-Order Logic

A) Reducing first-order inference to propositional inference
   - Removing $\forall$
   - Removing $\exists$
   - Unification

B) *Lifting* propositional inference to first-order inference
   - Generalized Modus Ponens
   - FOL forward chaining
Universal Instantiation

Every instantiation of a universally quantified sentence is entailed by it:

\[ \sqrt{\forall \nu \alpha} \]

\[ \text{Subst}\{\{\nu/g\}, \alpha\} \]

for any variable \( \nu \) and ground term \( g \)

E.g., \( \forall x \quad \text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x) \) yields

\( \text{King}(\text{John}) \land \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John}) \)
\( \text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard}) \)
\( \text{King}(\text{Father}(\text{John})) \land \text{Greedy}(\text{Father}(\text{John})) \Rightarrow \text{Evil}(\text{Father}(\text{John})) \)
Existential Instantiation

For any sentence $a$, variable $v$, and constant symbol $k$ that does not appear elsewhere in the knowledge base:

$$\exists v \ a$$
$$\text{Subst}\{\{v/k\}, a\}$$

E.g., $\exists x \ Crown(x) \land OnHead(x, John)$ yields
$$Crown(C_1) \land OnHead(C_1, John)$$

provided $C_1$ is a new constant symbol, called a **Skolem constant**
Suppose the KB contains just the following:

\[ \forall x \, \text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x) \]

King(John)
Greedy(John)
Brother(Richard, John)

Instantiating the universal sentence in all possible ways, we have

\[ \text{King}(\text{John}) \land \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John}) \]
\[ \text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard}) \]

King(John)
Greedy(John)
Brother(Richard, John)

The new KB is propositionalized: proposition symbols are

King(John), Greedy(John), Evil(John), King(Richard) etc.
Reduction to Propositional Inference

Claim: a ground sentence is entailed by new KB iff entailed by original KB

Claim: every FOL KB can be propositionalized so as to preserve entailment

Idea: propositionalize KB and query, apply resolution, return result

Problem: with function symbols, there are infinitely many ground terms,
e.g., Father(Father(Father(John)))

Theorem: Herbrand (1930). If a sentence \( \alpha \) is entailed by an FOL KB, it is entailed by a finite subset of the propositional KB

Idea: For \( n = 0 \) to \( \infty \) do

create a propositional KB by instantiating with depth-\( n \) terms see if \( \alpha \) is entailed by this KB

Problem: works if \( \alpha \) is entailed, loops if \( \alpha \) is not entailed

Theorem: Turing (1936), Church (1936), entailment in FOL is semidecidable
Problems with Propositionalization

Propositionalization seems to generate lots of irrelevant sentences. E.g., from

\[ \forall x \text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x) \]

\text{King(John)}

\text{Greedy(y)}

\text{Brother(Richard, John)}

it seems obvious that \text{Evil(John)}, but propositionalization produces lots of facts such as \text{Greedy(Richard)} that are irrelevant.
Unification

We can get the inference immediately if we can find a substitution $\theta$ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(y)$

$\theta = \{x/\text{John, } y/\text{John}\}$ works

Unify$(a, \beta) = \theta$ if $a\theta = \beta\theta$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows(John, x)</td>
<td>Knows(John, Jane)</td>
<td>${x/\text{Jane}}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, OJ)</td>
<td>${x/\text{OJ, } y/\text{John}}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, Mother(y))</td>
<td>${y/\text{John, } x/\text{Mother(John)}}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(x, OJ)</td>
<td>fail</td>
</tr>
</tbody>
</table>

Standardizing apart eliminates overlap of variables, e.g., $\text{Knows}(z_{17}, \text{OJ})$
Generalized Modus Ponens (GMP)

\[ p_1^t, \ p_2^t, \ldots, \ p_n^t, \ (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \]

\[ q^{\theta} \]

where \( p_i^{\theta} = p_i^{\theta} \) for all \( i \)

\( p_1^t \) is King(John)
\( p_2^t \) is Greedy(y)
\( \theta \) is \{x/John, y/John\}
\( q^{\theta} \) is Evil(John)

GMP used with KB of definite clauses (exactly one positive literal)
All variables assumed universally quantified
function FOL-FC-Ask(KB, α) returns a substitution or false

repeat until new is empty
    new ← { }
    for each sentence r in KB do
        (p₁ ∧ . . . ∧ pₙ ⇒ q) ← Standardize-Apart(r)
        for each θ such that (p₁ ∧ . . . ∧ pₙ)θ = (p₁ ∧ . . . ∧ pₙ)θₜ for some p₁ᵗ, . . . , pₙᵗ in KB
            qₜ ← Subst(θ, q)
            if qₜ is not a renaming of a sentence already in KB or new then do
                add qₜ to new
                φ ← Unify(qₜ, α)
                if φ is not fail then return φ
            end if
        end for
    end for
    add new to KB
return false