First Order Logic

CS171, Fall 2016
Introduction to Artificial Intelligence
Prof. Alexander Ihler
Outline

• New ontology
  – objects, relations, properties, functions.
• New Syntax
  – Constants, predicates, properties, functions
• New semantics
  – meaning of new syntax
• Inference rules for Predicate Logic (FOL)
  – Resolution
  – Forward-chaining, Backward-chaining
  – Unification
• Reading: Russell and Norvig Chapters 8 & 9
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
Building a more expressive language

Want to develop a better, more expressive language:

- Needs to refer to objects in the world,
- Needs to express general rules
  - On(x,y) \rightarrow \sim \text{clear}(y)
  - All men are mortal
  - Everyone over age 21 can drink
  - One student in this class got a perfect score
  - Etc....
- First order logic, or “predicate calculus” allows more expressiveness
## Logics in general

<table>
<thead>
<tr>
<th>Language</th>
<th>Ontological Commitment</th>
<th>Epistemological Commitment</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>Temporal logic</td>
<td>facts, objects, relations, times</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Probability theory</td>
<td>facts</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Fuzzy logic</td>
<td>degree of truth $\in [0, 1]$</td>
<td>degree of belief $\in [0, 1]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>known interval value</td>
</tr>
</tbody>
</table>
First-order logic

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, comes between, . . .
- **Functions**: father of, best friend, third inning of, one more than, beginning of . . .
Syntax of FOL: Basic elements

Constants  \textit{KingJohn, 2, UCB, \ldots}  
Predicates  \textit{Brother, >, \ldots}  
Functions  \textit{Sqrt, LeftLegOf, \ldots}  
Variables  \textit{x, y, a, b, \ldots}  
Connectives  \wedge \; \lor \; \neg \; \Rightarrow \; \Leftrightarrow  
Equality  =  
Quantifiers  \forall \; \exists
Atomic sentences

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

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

E.g., \( \text{Brother}(\text{KingJohn}, \text{RichardTheLionheart}) \)
\( > (\text{Length}(\text{LeftLegOf}(\text{Richard})), \text{Length}(\text{LeftLegOf}(\text{KingJohn}))) \)
Complex sentences

Complex sentences are made from atomic sentences using connectives

$$\neg S, \quad S_1 \land S_2, \quad S_1 \lor S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \iff S_2$$

E.g. $\text{Sibling}(\text{King John}, \text{Richard}) \Rightarrow \text{Sibling}(\text{Richard}, \text{King John})$

$$(1, 2) \lor (1, 2)$$

$$(1, 2) \land \neg (1, 2)$$
Semantics: Worlds

- The **world** consists of **objects** that have **properties**.
  - There are **relations** and **functions** between these objects.
  - Objects in the world, individuals: people, houses, numbers, colors, baseball games, wars, centuries
    - Clock A, John, 7, the-house in the corner, Tel-Aviv
  - Functions on individuals:
    - father-of, best friend, third inning of, one more than
  - Relations:
    - brother-of, bigger than, inside, part-of, has color, occurred after
  - Properties (a relation of arity 1):
    - red, round, bogus, prime, multistoried, beautiful
Semantics: Interpretation

• An interpretation of a sentence (wff) is an assignment that maps
  – Object constants to objects in the worlds,
  – n-ary function symbols to n-ary functions in the world,
  – n-ary relation symbols to n-ary relations in the world

• Given an interpretation, an atom has the value “true” if it denotes a relation that holds for those individuals denoted in the terms. Otherwise it has the value “false”
  – Example: Block world:
    • A,B,C,floor, On, Clear
  – World:
    – On(A,B) is false, Clear(B) is true, On(C,F1) is true...
Floor

B
A
C
Truth in first-order logic

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

- Model contains objects (domain elements) and relations among them.

- Interpretation specifies referents for:
  - Constant symbols → objects
  - Predicate symbols → relations
  - Function symbols → functional relations

- An atomic sentence \( \text{predicate}(term_1, \ldots, term_n) \) is true iff the objects referred to by \( term_1, \ldots, term_n \) are in the relation referred to by \( \text{predicate} \).
Semantics: Models

- An interpretation satisfies a wff (sentence) if the wff has the value “true” under the interpretation.
- Model: An interpretation that satisfies a wff is a model of that wff.
- Validity: Any wff that has the value “true” under all interpretations is valid.
- Any wff that does not have a model is inconsistent or unsatisfiable.
- If a wff w has a value true under all the models of a set of sentences KB then KB logically entails w.
Example of models (blocks world)

The formulas:
- On(A,F1) \implies Clear(B)
- Clear(B) and Clear(C) \implies On(A,F1)
- Clear(B) or Clear(A)
- Clear(B)
- Clear(C)

Possible interpretations that are models:

- On = \{<B,A>,<A,floor>,<C,floor>\}
- Clear = \{<C>,<B>\}
Models for FOL: Example

- person
- brother
- left leg
- crown on head
- person king
Quantification

• Universal and existential quantifiers allow expressing general rules with variables

• Universal quantification
  – All cats are mammals

\[ \forall x \text{Cat}(x) \rightarrow \text{Mammal}(x) \]

  – It is equivalent to the conjunction of all the sentences obtained by substitution the name of an object for the variable x.

• Syntax: if w is a wff then (forall x) w is a wff.

\[ \text{Cat}(\text{Spot}) \rightarrow \text{Mammal}(\text{Spot}) \wedge \\
\text{Cat}(\text{Rebbeka}) \rightarrow \text{Mammal}(\text{Rebbeka}) \wedge \\
\text{Cat}(\text{Felix}) \rightarrow \text{Mammal}(\text{Felix}) \wedge \\
\ldots \ldots \]
Quantification: Universal

• Universal quantification $\forall$: a universally quantified sentence is true if it is true for every object in the model
  Everyone in Irvine has a tan:

• Roughly equivalent to conjunction:
A common mistake

• Typically, “implies” = “⇒” is the main connective operator with ∀

• Everyone in Irvine has a tan:
  \[ ∀ \, x : \text{InIrvine}(x) ⇒ \text{Tan}(x) \]

• Operator ∧ is uncommon
  \[ ∀ \, x : \text{InIrvine}(x) ∧ \text{Tan}(x) \]
  means that everyone lives in Irvine and is tan.
Quantification: Existential

- Existential quantification \( \exists \) : an existentially quantified sentence is true in case one of the disjunct is true

  Spot has a sister who is a cat:

  \[ \exists x \text{Sister}(x, \text{spot}) \land \text{Cat}(x) \]

- Roughly equivalent to disjunction:

  \[
  \begin{align*}
  & \text{Sister} (\text{Spot}, \text{Spot}) \land \text{Cat} (\text{Spot}) \lor \\
  & \text{Sister} (\text{Rebecca}, \text{Spot}) \land \text{Cat} (\text{Rebecca}) \lor \\
  & \text{Sister} (\text{Felix}, \text{Spot}) \land \text{Cat} (\text{Felix}) \lor \\
  & \text{Sister} (\text{Richard}, \text{Spot}) \land \text{Cat} (\text{Richard})
  \end{align*}
  \]

- We can mix existential and universal quantification.
A common mistake

- Typically, “and” = “∧” is the main connective operator with ∃
- Spot has a sister who is a cat:
  \[ \exists x : \text{Sister}(x, \text{Spot}) \land \text{Cat}(x) \]

- Operator \( \Rightarrow \) is uncommon
  \[ \exists x : \text{Sister}(x, \text{Spot}) \Rightarrow \text{Cat}(x) \]
  is true if there is anyone who is not Spot’s sister
Properties of quantifiers

- $\forall x \forall y$ is the same as $\forall y \forall x$
- $\exists x \exists y$ is the same as $\exists y \exists x$
- $\exists x \forall y$ is not the same as $\forall y \exists x$

- $\exists x \forall y \text{ Loves}(x,y)$
  - “There is a person who loves everyone in the world”

- $\forall y \exists x \text{ Loves}(x,y)$
  - “Everyone in the world is loved by at least one person”

- Quantifier duality: each can be expressed using the other
  $\forall x \: \text{Likes}(x,\text{IceCream}) \iff \neg \exists x \: \neg \text{Likes}(x,\text{IceCream})$
  $\exists x \: \text{Likes}(x,\text{Broccoli}) \iff \neg \forall x \: \neg \text{Likes}(x,\text{Broccoli})$
Fun with sentences

Brothers are siblings
Fun with sentences

Brothers are siblings

∀ x, y  Brother(x, y) ⇒ Sibling(x, y).

“Sibling” is symmetric
Fun with sentences

Brothers are siblings
\[ \forall x, y \quad \text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y). \]

“Sibling” is symmetric
\[ \forall x, y \quad \text{Sibling}(x, y) \Leftrightarrow \text{Sibling}(y, x). \]

One’s mother is one’s female parent
Fun with sentences

Brothers are siblings

\[ \forall x, y \: \text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y). \]

“Sibling” is symmetric

\[ \forall x, y \: \text{Sibling}(x, y) \iff \text{Sibling}(y, x). \]

One’s mother is one’s female parent

\[ \forall x, y \: \text{Mother}(x, y) \iff (\text{Female}(x) \land \text{Parent}(x, y)). \]

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

• term1 = term2 is true under a given interpretation if and only if term1 and term2 refer to the same object

• E.g., definition of Sibling in terms of Parent:

$$\forall x, y \ \text{Sibling}(x,y) \iff 
\neg (x = y) \land \exists m, f \neg (m = f) \land \text{Parent}(m, x) \land \text{Parent}(f, x) \land \text{Parent}(m, y) \land \text{Parent}(f, y)$$
Using FOL

• The kinship domain:
  – object are people
  – Properties include gender and they are related by relations such as parenthood, brotherhood, marriage
  – predicates: Male, Female (unary) Parent, Sibling, Daughter, Son...
  – Function: Mother Father

• Brothers are siblings
  – \( \forall x, y \) Brother\((x, y) \Leftrightarrow\) Sibling\((x, y)\)

• One's mother is one's female parent
  – \( \forall m, c \) Mother\((c) = m \Leftrightarrow\) (Female\((m) \land\) Parent\((m, c))\)

• “Sibling” is symmetric
  – \( \forall x, y \) Sibling\((x, y) \Leftrightarrow Sibling(y, x)\)
Using FOL

- The set domain:
  - $\forall s \ Set(s) \iff (s = {} \lor (\exists x,s2 \ Set(s2) \land s = \{x | s2\})$
  - $\neg \exists x,s \ {x | s} = {}$
  - (Adjoining an element already in the set has no effect)
  - $\forall x,s \ x \in s \iff s = \{x | s\}$
  - (the only members of a set are the elements that were adjoined into it)
  - $\forall x,s \ x \in s \iff [\exists y,s2 \ (s = \{y | s2\} \land (x = y \lor x \in s2))]$
  - $\forall s1,s2 \ s1 \subseteq s2 \iff (\forall x \ x \in s1 \Rightarrow x \in s2)$
  - $\forall s1,s2 \ (s1 = s2) \iff (s1 \subseteq s2 \land s2 \subseteq s1)$
  - $\forall x,s1,s2 \ x \in (s1 \cap s2) \iff (x \in s1 \land x \in s2)$
  - $\forall x,s1,s2 \ x \in (s1 \cup s2) \iff (x \in s1 \lor x \in s2)$

**Objects** are sets

**Predicates:** unary predicate “set,” binary predicate membership ($x$ is a member of set), “subset” ($s1$ is a subset of $s2$)

**Functions:** intersections, union, adjoining an element to a set.
Knowledge engineering in FOL

- Identify the task
- Assemble the relevant knowledge
- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain
- Encode a description of the specific problem instance
- Pose queries to the inference procedure and get answers
- Debug the knowledge base
The electronic circuits domain

One-bit full adder
The electronic circuits domain

• Identify the task
  – Does the circuit actually add properly? (circuit verification)

• Assemble the relevant knowledge
  – Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
  – Irrelevant: size, shape, color, cost of gates

• Decide on a vocabulary
  – Alternatives:
    • \( \text{Type}(X1) = \text{XOR} \)
    • \( \text{Type}(X1, \text{XOR}) \)
    • \( \text{XOR}(X1) \)
The electronic circuits domain

• Encode general knowledge of the domain

  – ∀t1,t2 Connected(t1, t2) ⇒ Signal(t1) = Signal(t2)
  – ∀t Signal(t) = 1 v Signal(t) = 0
  – 1 ≠ 0

  – ∀t1,t2 Connected(t1, t2) ⇒ Connected(t2, t1)

  – ∀g Type(g) = OR ⇒ Signal(Out(1,g)) = 1 ⇔ ∃n Signal(In(n,g)) = 1
  – ∀g Type(g) = AND ⇒ Signal(Out(1,g)) = 0 ⇔ ∃n Signal(In(n,g)) = 0
  – ∀g Type(g) = XOR ⇒ Signal(Out(1,g)) = 1 ⇔ Signal(In(1,g)) ≠ Signal(In(2,g))
  – ∀g Type(g) = NOT ⇒ Signal(Out(1,g)) ≠ Signal(In(1,g))
The electronic circuits domain

• Encode the specific problem instance
  – Type(X1) = XOR       Type(X2) = XOR
  – Type(A1) = AND      Type(A2) = AND
  – Type(O1) = OR

  – Connected(Out(1,X1),In(1,X2))       Connected(In(1,C1),In(1,X1))
  – Connected(Out(1,X1),In(2,A2))       Connected(In(1,C1),In(1,A1))
  – Connected(Out(1,A2),In(1,O1))       Connected(In(2,C1),In(2,X1))
  – Connected(Out(1,A1),In(2,O1))       Connected(In(2,C1),In(2,A1))
  – Connected(Out(1,X2),Out(1,C1))      Connected(In(3,C1),In(2,X2))
  – Connected(Out(1,O1),Out(2,C1))      Connected(In(3,C1),In(1,A2))
The electronic circuits domain

6. Pose queries to the inference procedure

What are the possible sets of values of all the terminals for the adder circuit?

$$\exists i_1, i_2, i_3, o_1, o_2 \text{ Signal}(\text{In}(1, C_1)) = i_1 \land \text{Signal}(\text{In}(2, C_1)) = i_2 \land \text{Signal}(\text{In}(3, C_1)) = i_3 \land \text{Signal}(\text{Out}(1, C_1)) = o_1 \land \text{Signal}(\text{Out}(2, C_1)) = o_2$$

7. Debug the knowledge base

(May have omitted assertions like 1 ≠ 0)
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 $t = 5$:

$\text{Tell}(KB, \text{Percept}([\text{Smell}, \text{Breeze}, \text{None}], 5))$
$\text{Ask}(KB, \exists a \ \text{Action}(a, 5))$

I.e., does the KB entail any particular actions at $t = 5$?

Answer: $Yes$, \{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.,
$S = \text{Smarter}(x, y)$
$\sigma = \{x/\text{Hillary}, y/\text{Bill}\}$
$S\sigma = \text{Smarter}(\text{Hillary}, \text{Bill})$

$\text{Ask}(KB, S)$ returns some/all $\sigma$ such that $KB \models S\sigma$
Knowledge base for the wumpus world

“Perception”
\[ \forall b, g, t \ Percept([Smell, b, g], t) \Rightarrow Smell(t) \]
\[ \forall s, b, t \ Percept([s, b, Glitter], t) \Rightarrow AtGold(t) \]

**Reflex:** \[ \forall t \ AtGold(t) \Rightarrow Action(Grab, t) \]

**Reflex with internal state:** do we have the gold already?
\[ \forall t \ AtGold(t) \land \neg Holding(Gold, t) \Rightarrow Action(Grab, t) \]

*Holding(Gold, t)* cannot be observed
\[ \Rightarrow \text{keeping track of change is essential} \]
Deducing hidden properties

Properties of locations:
∀ x, t  At(Agent, x, t) ∧ Smelt(t) ⇒ Smelly(x)
∀ x, t  At(Agent, x, t) ∧ Breeze(t) ⇒ Breezy(x)

Squares are breezy near a pit:

**Diagnostic** rule—infer cause from effect
∀ y  Breezy(y) ⇒ ∃ x  Pit(x) ∧ Adjacent(x, y)

**Causal** rule—infer effect from cause
∀ x, y  Pit(x) ∧ Adjacent(x, y) ⇒ 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:
∀ y  Breezy(y) ⇔ [∃ x  Pit(x) ∧ Adjacent(x, y)]
Keeping track of change

Facts hold in situations, rather than eternally
E.g., $\text{Holding}(\text{Gold}, \text{Now})$ rather than just $\text{Holding}(\text{Gold})$

Situation calculus is one way to represent change in FOL:
  Adds a situation argument to each non-eternal predicate
E.g., $\text{Now}$ in $\text{Holding}(\text{Gold}, \text{Now})$ 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 I

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

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

Frame problem: find an elegant way to handle non-change
   (a) representation—avoid frame axioms
   (b) inference—avoid repeated “copy-overs” to keep track of state

Qualification problem: true descriptions of real actions require endless caveats—what if gold is slippery or nailed down or . . .

Ramification problem: real actions have many secondary consequences—what about the dust on the gold, wear and tear on gloves, . . .
Describing actions II

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:

\[ \forall a, s \; Holding(\text{Gold}, Result(a, s)) \iff \]
\[ [(a = \text{Grab} \land AtGold(s)) \]
\[ \lor (Holding(\text{Gold}, s) \land a \neq \text{Release})] \]
Some more notation

• Instantiation: specify values for variables

• Ground term
  – A term without variables

• Substitution
  – Setting a variable equal to something
  – \( \theta = \{x / \text{John}, y / \text{Richard}\} \)
  – Read as “\(x := \text{John}, y:=\text{Richard}\)”

• Write a substitution into sentence \(\alpha\) as
  \(\text{Subst}(\theta, \alpha)\) or just as \(\alpha\theta\)
Universal instantiation (UI)

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

\[ \forall v \alpha \]
\[ \text{Subst}\{\{v/g\}, \alpha\} \]

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

• E.g., \( \forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x) \) yields:
  \( King(John) \land Greedy(John) \Rightarrow Evil(John) \)
  \( King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard) \)
  \( King(Father(John)) \land Greedy(Father(John)) \Rightarrow Evil(Father(John)) \)
Existential instantiation (EI)

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

\[
\exists v \alpha \quad \text{Subst}\{v/k\}, \alpha
\]

• E.g., $\exists x \text{Crown}(x) \land \text{OnHead}(x, John)$ yields:

$\text{Crown}(C_1) \land \text{OnHead}(C_1, John)$

provided $C_1$ is a new constant symbol, called a Skolem constant
Reduction to propositional inference

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:

King(John) \land \text{ Greedy}(John) \Rightarrow \text{ Evil}(John)
King(Richard) \land \text{ Greedy}(Richard) \Rightarrow \text{ Evil}(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 contd.

• Every FOL KB can be propositionalized so as to preserve entailment

• (A ground sentence is entailed by new KB iff entailed by original KB)

• **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 propositionalized 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 for FOL is semidecidable (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence.)
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}(\text{John})
  \forall y \text{ Greedy}(y)
  \text{ Brother}(\text{Richard}, \text{John})

- Given query “\(\text{evil}(x)\) it seems obvious that \(\text{Evil}(\text{John})\), but propositionalization produces lots of facts such as \(\text{Greedy}(\text{Richard})\) that are irrelevant.

- With \(p\) \(k\)-ary predicates and \(n\) constants, there are \(p \cdot n^k\) instantiations.
Generalized Modus Ponens (GMP)

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

\[ q \theta \]

where \( p_i \theta = p_i \theta \) for all \( i \)

- \( p_1' \) is \( King(John) \)
- \( p_1 \) is \( King(x) \)
- \( p_2' \) is \( Greedy(y) \)
- \( p_2 \) is \( Greedy(x) \)
- \( \theta \) 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
Soundness of GMP

- Need to show that
  
  \[ p_1', \ldots, p_n', (p_1 \land \ldots \land p_n \Rightarrow q) \models q_\theta \]

  provided that \( p_i' \theta = p_i \theta \) for all \( i \)

- Lemma: For any sentence \( p \), we have \( p \models p_\theta \) by UI

1. \( (p_1 \land \ldots \land p_n \Rightarrow q) \models (p_1 \land \ldots \land p_n \Rightarrow q)_\theta = (p_1 \theta \land \ldots \land p_n \theta \Rightarrow q_\theta) \)

2. \( p_1', \ldots, \land; p_n' \models p_1' \land \ldots \land p_n' \models p_1 \theta \land \ldots \land p_n \theta \)

3. From 1 and 2, \( q_\theta \) follows by ordinary Modus Ponens
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

- $\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

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

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

- \( \text{Unify}(\alpha, \beta) = \theta \) if \( \alpha \theta = \beta \theta \)

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

- Standardizing apart eliminates overlap of variables, e.g., \( \text{Knows}(z_{17}, \text{OJ}) \).
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

- $\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

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

- **Standardizing apart** eliminates overlap of variables, e.g., $\text{Knows}(z_{17}, \text{OJ})$
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

- \( \text{Unify}(\alpha, \beta) = \theta \) if \( \alpha \theta = \beta \theta \)

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

- Standardizing apart eliminates overlap of variables, e.g., \( \text{Knows}(z_{17},\text{OJ}) \)
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

- $\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

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

- Standardizing apart eliminates overlap of variables, e.g., $\text{Knows}(z_{17},\text{OJ})$
Unification

- To unify $\text{Knows}(John,x)$ and $\text{Knows}(y,z)$,

\[ \theta = \{y/John, x/z\} \text{ or } \theta = \{y/John, x/John, z/John\} \]

- The first unifier is more general than the second.

- There is a single most general unifier (MGU) that is unique up to renaming of variables.

\[ \text{MGU} = \{y/John, x/z\} \]
The unification algorithm

function UNIFY(x, y, \( \theta \)) returns a substitution to make \( x \) and \( y \) identical

inputs: \( x \), a variable, constant, list, or compound
\( y \), a variable, constant, list, or compound
\( \theta \), the substitution built up so far

if \( \theta = \text{failure} \) then return failure
else if \( x = y \) then return \( \theta \)
else if VARIABLE?(\( x \)) then return UNIFY-VAR(\( x, y, \theta \))
else if VARIABLE?(\( y \)) then return UNIFY-VAR(\( y, x, \theta \))
else if COMPOUND?(\( x \)) and COMPOUND?(\( y \)) then
    return UNIFY(ARGS[\( x \)], ARGSS[\( y \)], UNIFY(\( \text{Op}[x], \text{Op}[y], \theta \)))
else if LIST?(\( x \)) and LIST?(\( y \)) then
    return UNIFY(REST[\( x \)], REST[\( y \)], UNIFY(FIRST[\( x \)], FIRST[\( y \)], \( \theta \))
else return failure
The unification algorithm

function UNIFY-VAR(var, x, θ) returns a substitution
inputs: var, a variable
        x, any expression
        θ, the substitution built up so far

if \{var/val\} ∈ θ then return UNIFY(val, x, θ)
else if \{x/val\} ∈ θ then return UNIFY(var, val, θ)
else if OCCUR-CHECK?(var, x) then return failure
else return add \{var/x\} to θ
Example knowledge base

• The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

• Prove that Col. West is a criminal
... it is a crime for an American to sell weapons to hostile nations:
\[
\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)
\]

Nono ... has some missiles, i.e., \( \exists x \text{ Owns}(\text{Nono},x) \land \text{Missile}(x) \):
\[
\text{Owns}(\text{Nono},M_1) \text{ and } \text{Missile}(M_1)
\]

... all of its missiles were sold to it by Colonel West
\[
\text{Missile}(x) \land \text{Owns}(\text{Nono},x) \Rightarrow \text{Sells}(\text{West},x,\text{Nono})
\]

Missiles are weapons:
\[
\text{Missile}(x) \Rightarrow \text{Weapon}(x)
\]

An enemy of America counts as "hostile“:
\[
\text{Enemy}(x,\text{America}) \Rightarrow \text{Hostile}(x)
\]

West, who is American ...
\[
\text{American}(\text{West})
\]

The country Nono, an enemy of America ...
\[
\text{Enemy}(\text{Nono},\text{America})
\]
Forward chaining algorithm

```latex
function FOL-FC-Ask(KB, \alpha) \text{ returns a substitution or } false
    \textbf{repeat until } new \textbf{ is empty}
    \hspace{1em} new \leftarrow \{ \}
    \textbf{for each sentence } r \textbf{ in } KB \textbf{ do}
    \hspace{2em} (p_1 \land \ldots \land p_n \Rightarrow q) \leftarrow \text{STANDARDIZE-APART}(r)
    \hspace{2em} \textbf{for each } \theta \textbf{ such that } (p_1 \land \ldots \land p_n)\theta = (p'_1 \land \ldots \land p'_n)\theta
    \hspace{3em} \textbf{for some } p'_1, \ldots, p'_n \textbf{ in } KB
    \hspace{4em} q' \leftarrow \text{SUBST}(\theta, q)
    \hspace{4em} \textbf{if } q' \textbf{ is not a renaming of a sentence already in } KB \textbf{ or } new \textbf{ then do}
    \hspace{5em} \text{add } q' \textbf{ to } new
    \hspace{5em} \phi \leftarrow \text{UNIFY}(q', \alpha)
    \hspace{6em} \textbf{if } \phi \textbf{ is not fail then return } \phi
    \hspace{4em} \text{add } new \textbf{ to } KB
    \hspace{2em} \textbf{return } false
```
Forward chaining proof
Forward chaining proof

\[ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]

\[ \text{Missile}(x) \Rightarrow \text{Weapon}(x) \]

\[ \text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x) \]
Forward chaining proof

\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]
Forward chaining proof

*American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)
*Owns(Nono,M1) and Missile(M1)
*Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)
*Missile(x) ⇒ Weapon(x)
*Enemy(x,America) ⇒ Hostile(x)
*American(West)
*Enemy(Nono,America)
Properties of forward chaining

• Sound and complete for first-order definite clauses

• **Datalog** = first-order definite clauses + no functions

• FC terminates for Datalog in finite number of iterations

• May not terminate in general if \( \alpha \) is not entailed

• This is unavoidable: entailment with definite clauses is semidecidable
Efficiency of forward chaining

Incremental forward chaining: no need to match a rule on iteration \( k \) if a premise wasn't added on iteration \( k-1 \)

\[ \Rightarrow \text{match each rule whose premise contains a newly added positive literal} \]

Matching itself can be expensive:

Database indexing allows \( O(1) \) retrieval of known facts

- e.g., query \( \text{Missile}(x) \) retrieves \( \text{Missile}(M_1) \)

Forward chaining is widely used in deductive databases
Hard matching example

- **Colorable()** is inferred iff the CSP has a solution
- CSPs include 3SAT as a special case, hence matching is NP-hard

\[
\text{Diff}(wa, nt) \land \text{Diff}(wa, sa) \land \text{Diff}(nt, q) \land \\
\text{Diff}(nt, sa) \land \text{Diff}(q, nsw) \land \text{Diff}(q, sa) \land \\
\text{Diff}(nsw, v) \land \text{Diff}(nsw, sa) \land \text{Diff}(v, sa) \Rightarrow \\
\text{Colorable()}
\]

\[
\text{Diff}(\text{Red}, \text{Blue}) \quad \text{Diff}(\text{Red}, \text{Green}) \\
\text{Diff}(\text{Green}, \text{Red}) \quad \text{Diff}(\text{Green}, \text{Blue}) \\
\text{Diff}(\text{Blue}, \text{Red}) \quad \text{Diff}(\text{Blue}, \text{Green})
\]
Backward chaining example

Criminal(West)
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining algorithm

function FOL-BC-Ask(KB, goals, θ) returns a set of substitutions
  inputs: KB, a knowledge base
           goals, a list of conjuncts forming a query
           θ, the current substitution, initially the empty substitution { }  
  local variables: ans, a set of substitutions, initially empty
  if goals is empty then return {θ}
  q' ← SUBST(θ, FIRST(goals))
  for each r in KB where STANDARDIZE-Apart(r) = (p₁ ∧ ... ∧ pₙ ⇒ q)
                          and θ' ← UNIFY(q, q') succeeds
  ans ← FOL-BC-Ask(KB, [p₁, ..., pₙ|REST(goals)], COMPOSE(θ, θ')) ∪ ans
  return ans

SUBST(COMPOSE(θ₁, θ₂), p) = SUBST(θ₂, SUBST(θ₁, p))
Properties of backward chaining

• Depth-first recursive proof search: space is linear in size of proof

• Incomplete due to infinite loops
  → fix by checking current goal against every goal on stack

• Inefficient due to repeated subgoals (both success and failure)
  → fix using caching of previous results (extra space)

• Widely used for logic programming
Logic programming: Prolog

• Algorithm = Logic + Control

• Basis: backward chaining with Horn clauses + bells & whistles
  Widely used in Europe, Japan (basis of 5th Generation project)
  Compilation techniques ⇒ 60 million LIPS

• Program = set of clauses = head :- literal₁, ... literalₙ.

  criminal(X) :- american(X), weapon(Y), sells(X,Y,Z), hostile(Z).

• Depth-first, left-to-right backward chaining
• Built-in predicates for arithmetic etc., e.g., X is Y*Z+3
• Built-in predicates that have side effects (e.g., input and output

• predicates, assert/retract predicates)
• Closed-world assumption ("negation as failure")
  – e.g., given alive(X) :- not dead(X).
  – alive(joe) succeeds if dead(joe) fails
Prolog

• Appending two lists to produce a third:

\[
\text{append}([], Y, Y). \\
\text{append}([X|L], Y, [X|Z]) :- \text{append}(L, Y, Z).
\]

• query: \( \text{append}(A, B, [1,2]) \) ?

• answers: \( A = [] \quad B = [1,2] \)


\( A = [1,2] \quad B = [] \)
Resolution: brief summary

• Full first-order version:

\[
\frac{\ell_1 \lor \cdots \lor \ell_k \lor m_1 \lor \cdots \lor m_n}{(\ell_1 \lor \cdots \lor \ell_{i-1} \lor \ell_{i+1} \lor \cdots \lor \ell_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n)\theta}
\]

where \( \text{Unify}(\ell_i, \neg m_j) = \theta \).

• The two clauses are assumed to be standardized apart so that they share no variables.

• For example,

\[
\neg \text{Rich}(x) \lor \text{Unhappy}(x)
\]

\[
\begin{array}{c}
\text{Rich}(\text{Ken}) \\
\hline
\text{Unhappy}(\text{Ken})
\end{array}
\]

with \( \theta = \{x/\text{Ken}\} \)

• Apply resolution steps to \( \text{CNF}(KB \land \neg \alpha) \); complete for FOL
Conversion to CNF

• Everyone who loves all animals is loved by someone:
  \( \forall x [\forall y \ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y Loves(y,x)] \)

1. Eliminate biconditionals and implications
   \( \forall x [\neg \forall y \neg Animal(y) \lor Loves(x,y)] \lor [\exists y Loves(y,x)] \)

2. Move \( \neg \) inwards:
   \( \neg \forall x \ p \equiv \exists x \ \neg p, \ \neg \exists x \ p \equiv \forall x \ \neg p \)
   \( \forall x [\exists y \neg (\neg Animal(y) \lor Loves(x,y))] \lor [\exists y Loves(y,x)] \)
   \( \forall x [\exists y \neg Animal(y) \land \neg Loves(x,y)] \lor [\exists y Loves(y,x)] \)
   \( \forall x [\exists y Animal(y) \land \neg Loves(x,y)] \lor [\exists y Loves(y,x)] \)
Conversion to CNF contd.

3. Standardize variables: each quantifier should use a different one

\[ \forall x \left[ \exists y \ Animal(y) \land \neg Loves(x,y) \right] \lor \left[ \exists z \ Loves(z,x) \right] \]

4. Skolemize: a more general form of existential instantiation. Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:

\[ \forall x \left[ Animal(F(x)) \land \neg Loves(x,F(x)) \right] \lor Loves(G(x),x) \]

5. Drop universal quantifiers:

\[ \left[ Animal(F(x)) \land \neg Loves(x,F(x)) \right] \lor Loves(G(x),x) \]

6. Distribute \(\lor\) over \(\land\):

\[ \left[ Animal(F(x)) \lor Loves(G(x),x) \right] \land \left[ \neg Loves(x,F(x)) \lor Loves(G(x),x) \right] \]
Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:
\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \implies \text{Criminal}(x) \]

Nono ... has some missiles, i.e., \( \exists x \ \text{Owns}(\text{Nono},x) \land \text{Missile}(x) \):
\[ \text{Owns}(\text{Nono},M) \land \text{Missile}(M) \]

... all of its missiles were sold to it by Colonel West
\[ \text{Missile}(x) \land \text{Owns}(\text{Nono},x) \implies \text{Sells}(\text{West},x,\text{Nono}) \]

Missiles are weapons:
\[ \text{Missile}(x) \implies \text{Weapon}(x) \]

An enemy of America counts as "hostile“:
\[ \text{Enemy}(x,\text{America}) \implies \text{Hostile}(x) \]

West, who is American ...
\[ \text{American}(\text{West}) \]

The country Nono, an enemy of America ...
\[ \text{Enemy}(\text{Nono},\text{America}) \]
Resolution proof: definite clauses

\[ \neg \text{American}(x) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(x,y,z) \lor \neg \text{Hostile}(z) \lor \text{Criminal}(x) \]

1. \text{American}(West)
2. \text{Missile}(x) \lor \text{Weapon}(x)
3. \text{Missile}(M1)
4. \text{Missile}(x) \lor \neg \text{Owns}(Nono,x) \lor \text{Sells}(West,x,Nono)
5. \text{Missile}(M1)
6. \text{Owns}(Nono,M1)
7. \neg \text{Enemy}(x,\text{America}) \lor \text{Hostile}(x)
8. \neg \text{Enemy}(Nono,\text{America})
Converting to clause form

\[ \forall x, y \: P(x) \land P(y) \land I(x,27) \land I(y,28) \rightarrow S(x, y) \]
[\[ P(A), P(B) \]
[\[ I(A,27) \lor I(A,28) \]
[\[ I(B,27) \]
[\[ \neg S(B, A) \]

Prove \( I(A,27) \)
Example: Resolution Refutation Prove $I(A, 27)$

\[ \neg I(A, 27) \]
\[ I(A, 27) \lor I(A, 28) \]
\[ \neg P(x) \lor \neg P(y) \lor \neg I(x, 27) \lor \neg I(y, 28) \lor S(x, y) \]
\[ I(A, 28) \]
\[ \neg P(x) \lor \neg P(A) \lor \neg I(x, 27) \]
\[ \lor S(x, A) \]
\[ \{A/y\} \]
\[ P(A) \]
\[ \neg P(x) \lor \neg I(x, 27) \lor S(x, A) \]
\[ I(B, 27) \]
\[ \neg I(B, 27) \lor S(B, A) \]
\[ \{B/x\} \]
\[ \neg S(B, A) \]
\[ S(B, A) \]
\[ \text{Nil} \]
Example: Answer Extraction

\[ \neg I(A, u) \lor \text{Ans}(u) \]
\[(\text{negation of wff to be proved with answer literal})\]

\[ I(A, 28) \lor \text{Ans}(27) \]
\[ I(A, 27) \lor I(A, 28) \]

\[ \neg P(x) \lor \neg P(y) \lor \neg I(x, 27) \lor \neg I(y, 28) \lor S(x, y) \]

\[ I(A, 28) \lor \text{Ans}(27) \]
\[ \{27/u\} \]

\[ \neg P(x) \lor \neg P(y) \lor \neg I(x, 27) \]
\[ \lor S(x, A) \lor \text{Ans}(27) \]
\[ \{A/y\} \]

\[ \neg P(x) \lor \neg I(x, 27) \lor S(x, A) \lor \text{Ans}(27) \]
\[ \{B/x\} \]

\[ I(B, 27) \]

\[ \neg I(B, 27) \lor S(B, A) \lor \text{Ans}(27) \]
\[ \{B/x\} \]

\[ \neg S(B, A) \]

\[ S(B, A) \lor \text{Ans}(27) \]

\[ \text{Ans}(27) \]