First-order Logic

Bùi Tiến Lên

01/09/2019

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Contents

1. Representation Revisited
2. Syntax and Semantics of First-Order Logic (FOL)
3. Using First-Order Logic
4. Propositional vs. First-Order Inference
5. Unification and Lifting
6. Forward Chaining
7. Backward Chaining
8. Resolution

Representation Revisited

Programming languages

- **Programming language** is a kind of formal languages. Some common programming languages are C++, Java or Lisp, etc.
- **Programs** represent computational processes while their **data structures** represent facts.
  - E.g., the Wumpus world can be represented by a 4×4 array, "World[2,2] $\leftarrow$ Pit" states that "There is a pit in square [2,2]."
- **Lack of general mechanisms** to derive facts from other facts
  - Update to a data structure is done by a domain-specific procedure.
- **Lack of expressiveness** to handle partial information
  - E.g., to say "There is a pit in [2,2] or [3,1]", a program stores a single value for each variable and allows the value to be "unknown", while the propositional logic sentence, $P_{2,2} \vee P_{1,1}$, is more intuitive.

Propositional logic

- ☺ **Propositional logic** is a **declarative language**.
  - Semantics is based on the truth relation between sentences and possible worlds.
- ☺ Propositional logic allows partial/disjunctive/negated information
  - Unlike most data structures and databases
- ☺ Propositional logic is **compositional**, which is desirable in representation languages
  - The meaning of a sentence is a function of the meaning of its parts; e.g., the meanings of $S_{1,4} \wedge S_{1,2}$ relates the meanings of $S_{1,4}$ and $S_{1,2}$.
- ☺ Meaning in propositional logic is **context-independent**
  - Unlike natural language, where meaning depends on context
- ☹ Propositional logic has very limited expressive power
  - E.g., cannot say "pits cause breezes in adjacent squares"

First-order logic

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

- **Objects**: are referred by nouns and noun phrases
  - E.g., people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries ...
- **Relations**: can be unary relations (properties) or n-ary relations, representing by verbs and verb phrases
  - Properites: red, round, bogus, prime, multistoried, etc.
  - n-ary relations: brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, etc.
- **Functions**: are relations in which there is only one "value" for a given "input."
  - E.g., father of, best friend, third inning of, one more than, etc.

Logics in general

Language	**Ontological Commitment** (What exists in the world)	**Epistemological Commitment** (What an agent believes about facts)
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, time	true/false/unknown
Probability logic	facts	degree of belief $\in [0, 1]$
Fuzzy logic	facts + degree of truth $\in [0, 1]$	known interval value

Syntax and Semantics of First-Order Logic (FOL)

BNF Grammar

$$
\begin{array}{rcl}
\mathit{Sentence} & \rightarrow & \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} & \rightarrow & \mathit{Predicate} \mid \mathit{Predicate}(\mathit{Term}, \ldots) \mid \mathit{Term}_1 = \mathit{Term}_2 \\
\mathit{ComplexSentence} & \rightarrow & (\mathit{Sentence}) \mid [\mathit{Sentence}] \\
& \mid & \neg \mathit{Sentence} \\
& \mid & \mathit{Sentence} \wedge \mathit{Sentence} \\
& \mid & \mathit{Sentence} \vee \mathit{Sentence} \\
& \mid & \mathit{Sentence} \implies \mathit{Sentence} \\
& \mid & \mathit{Sentence} \iff \mathit{Sentence} \\
& \mid & \mathit{Quantifier}\ \mathit{Variable}, \ldots \mathit{Sentence} \\
\\
\mathit{Term} & \rightarrow & \mathit{Function}(\mathit{Term}, \ldots) \\
& \mid & \mathit{Constant} \\
& \mid & \mathit{Variable} \\
\\
\mathit{Quantifier} & \rightarrow & \forall \mid \exists \\
\mathit{Constant} & \rightarrow & A \mid X_1 \mid \mathit{John} \mid \ldots \\
\mathit{Variable} & \rightarrow & a \mid x \mid s \mid \ldots \\
\mathit{Predicate} & \rightarrow & \mathit{True} \mid \mathit{False} \mid \mathit{After} \mid \mathit{Loves} \mid \mathit{Raining} \mid \ldots \\
\mathit{Function} & \rightarrow & \mathit{Mother} \mid \mathit{LeftLeg} \mid \ldots \\
\\
\textsc{Operator Precedence} & : & \neg, =, \wedge, \vee, \implies, \iff
\end{array}
$$

Models for First-order logic

- Models for first-order logic are more interesting with objects.
- The domain of a model is the set of objects (or domain elements) it contains.
- Nonempty:
  - Every possible world must contain at least one object
  - It doesn't matter what these objects are but how many there are in each particular model.

Models for FOL: Example

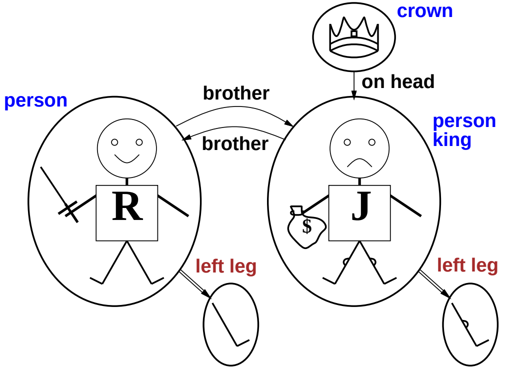


Figure 1: A model containing five objects, two binary relations, three unary relations (indicated by labels on the objects), and one unary function, left-leg.

Possible models in First-order logic

- Similar to propositional logic, entailment, validity, and so on are defined in terms of all possible models.
- The number of possible models is unbounded $\rightarrow$ checking entailment by the enumeration is **infeasible**.

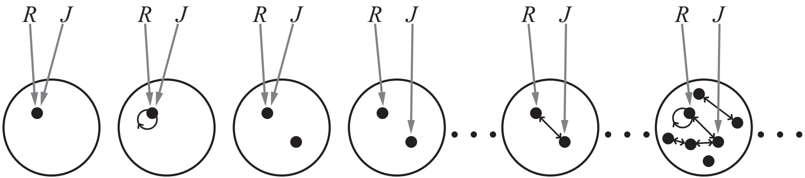


Figure 2: 137,506,194,466 models with six or fewer objects.

Quantifiers

Concept 1 (Universal quantification)

$\forall \langle \textit{variables} \rangle \ \langle \textit{sentence} \rangle$

$\forall x\ P$ is true in a model m iff P is true with x being *each* possible object in the model

- "Everyone at Berkeley is smart"

$$\forall x\ At(x, Berkeley) \implies Smart(x)$$

equivalent to the **conjunction** of **instantiations** of P

$$\begin{aligned}
&(At(KingJohn, Berkeley) \implies Smart(KingJohn))\\
&\wedge(At(Richard, Berkeley) \implies Smart(Richard))\\
&\wedge(At(Berkeley, Berkeley) \implies Smart(Berkeley))\\
&\wedge \dots
\end{aligned}$$

Quantifiers (cont.)

Concept 2 (Existential quantification)

$\exists \langle variables \rangle \ \langle sentence \rangle$

$\exists x\ P$ is true in a model m iff P is true with x being *some* possible object in the model

- "Someone at Stanford is smart"

$$\exists x\ At(x, Stanford) \wedge Smart(x)$$

equivalent to the **disjunction** of **instantiations** of P

$$\begin{aligned}
&(At(KingJohn, Berkeley) \implies Smart(KingJohn))\\
&\vee(At(Richard, Berkeley) \implies Smart(Richard))\\
&\vee(At(Berkeley, Berkeley) \implies Smart(Berkeley))\\
&\vee \ldots
\end{aligned}$$

A common mistake to avoid

- Typically, $\implies$ is the main connective with $\forall$
  - Common mistake: using $\wedge$ as the main connective with $\forall$
- Typically, $\wedge$ is the main connective with $\exists$
  - Common mistake: using $\implies$ as the main connective with $\exists$
- $\forall x\ At(x, Berkeley) \wedge Smart(x)$ means "Everyone is at Berkeley and everyone is smart"
  - **Too strong** implication
- $\exists x\ At(x, Stanford) \implies Smart(x)$ means "It is true even with anyone who is not at Stanford"
  - **Too weak** implication

Nested quantifiers

Multiple quantifiers enable more complex sentences.

- Simplest cases: Quantifiers are of the same type
  $\forall x \forall y \; Brother(x, y) \implies Sibling(x, y)$
  $\forall x \forall y \; Sibling(x, y) \iff Sibling(x, y)$
- Mixtures
  $\forall x \exists y \; Loves(x, y) \rightarrow$ "Everybody loves somebody"
  $\exists x \forall y \; Loves(x, y) \rightarrow$ "There is someone loved by everyone"
- **The order** of quantification is therefore very important.

Nested quantifiers (cont.)

Confusion: can arise when two quantifiers are used with the same variable name

$$\forall x\ (Crown(x) \vee (\exists x\ Brother(Richard, x)))$$

- **Rule**: The variable belongs to the **innermost** quantifier that mentions it.
- **Workaround**: Use different variable names with nested quantifier

$$\forall x\ (Crown(x) \vee (\exists z\ Brother(Richard, z)))$$

Properties of quantifiers

- Nested quantifiers

$$\begin{aligned}\forall x \forall y\, P &\equiv \forall y \forall x\, P\\ \exists x \exists y\, P &\equiv \exists y \exists x\, P\\ \exists x \forall y\, P &\not\equiv \forall y \exists x\, P\end{aligned}$$

- De Morgan's rules

$$\begin{aligned}\forall x\, \neg P &\equiv \neg \exists x\, P\\ \neg \forall x\, P &\equiv \exists x\, \neg P\\ \forall x\, P &\equiv \neg \exists x\, \neg P\\ \neg \exists x\, \neg P &\equiv \exists x\, P\end{aligned}$$

Fun with sentences

- Brothers are siblings
  - $\forall x, y\ Brother(x, y) \implies Sibling(x, y).$
- "Sibling" is symmetric
  - $\forall x, y\ Sibling(x, y) \iff Sibling(y, x).$
- One's mother is one's female parent
  - $\forall x, y\ Mother(x, y) \iff (Female(x) \wedge Parent(x, y)).$
- A first cousin is a child of a parent's sibling
  - $\forall x, y\ FirstCousin(x, y) \iff$
    $\exists p, ps\ Parent(p, x) \wedge Sibling(ps, p) \wedge Parent(ps, y)$

Equality

Concept 3

$term_1 = term_2$ is true under a given interpretation if and only if $term_1$ and $term_2$ refer to the same object

- $\neg(term_1 = term_2)$ means $term_1$ and $term_2$ not refer to the same object (sometimes write as $term_1 \neq term_2$)

- $Father(John) = Henry$ means that $Father(John)$ and $Henry$ refer to the same object
- Definition of (full) *Sibling* in terms of *Parent*:

$$\begin{aligned}\forall x, y\ Sibling(x, y) \quad \Longleftrightarrow \quad & \neg(x = y) \wedge \\ & \exists m, f\ \neg(m = f) \wedge \\ & Parent(m, x) \wedge Parent(f, x) \wedge \\ & Parent(m, y) \wedge Parent(f, y)\end{aligned}$$

Using First-Order Logic

Using First-Order Logic

- In knowledge representation, a **domain** is just some part of the world about which we wish to express some knowledge.
- First-order knowledge base KB has $\textsc{Tell}/\textsc{Ask}/\textsc{AskVars}$ interface
- Sentences (**assertions**) are added to a knowledge base KB using $\textsc{Tell}$

  $\textsc{Tell}(KB, King(John))$
  $\textsc{Tell}(KB, Person(Richard))$
  $\textsc{Tell}(KB, \forall x\ King(x) \implies Person(x))$

- We can ask questions (**queries** or **goals**) of the knowledge base KB using $\textsc{Ask}$

  $\textsc{Ask}(KB, Person(John)) \rightarrow$ *return true*
  $\textsc{Ask}(KB, \exists x\ Person(x)) \rightarrow$ *return true*

- If we want to know what value of x makes the sentence true using $\textsc{AskVars}$

  $\textsc{AskVars}(KB, Person(x)) \rightarrow$ *return a* ***substitution*** *list* $\{x/John\}$ *and* $\{x/Richard\}$

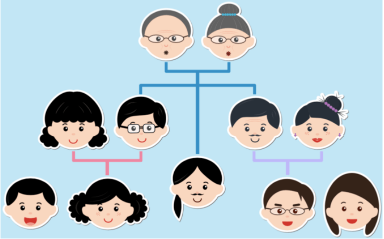
Using First-Order Logic (cont.)

- The assertions can be considered as the **axioms**
- Logical sentences which are entailed by the axioms are called **theorems**
- The theorems do not increase the set of conclusions that follow from the knowledge base KB.

  From a practical point of view, theorems are essential to reduce the computational cost of deriving new sentences

The Kinship Domain

- Unary predicates
  - Male and Female
- Binary predicates represent kinship relations
  - Parenthood, brotherhood, marriage, etc.
  - Parent, Sibling, Brother , Sister, Child, Daughter, Son, Spouse, Wife, Husband, Grandparent , Grandchild , Cousin, Aunt, and Uncle.
- Functions
  - Mother and Father, each person has exactly one of each of these.



The Little Kinship Domain

The possible axioms for Kinship domain

1. One's mother is one's female parent

$$\forall m, c \; Mother(c) = m \iff Female(m) \land Parent(m, c).$$

2. One's husband is one's male spouse

$$\forall w, h \; Husband(h, w) \iff Male(h) \land Spouse(h, w).$$

3. Male and female are disjoint categories

$$\forall x \; Male(x) \iff \neg Female(x).$$

4. Parent and child are inverse relations

$$\forall p, c \; Parent(p, c) \iff Child(c, p).$$

The Little Kinship Domain (cont.)

5. A grandparent is a parent of one's parent

$$\forall g, c\; Grandparent(g, c) \iff \exists p\; Parent(g, p) \wedge Parent(p, c).$$

6. A sibling is another child of one's parents

$$\forall x, y\; Sibling(x, y) \iff x \neq y \wedge \exists p\; Parent(p, x) \wedge Parent(p, y).$$

Using axioms to entail theorems

$$\text{axioms of kinship} \models \forall x \forall y\; Sibling(x, y) \iff Sibling(y, x)$$

Representation Revisited

Syntax and Semantics of First-Order Logic (FOL)

Using First-Order Logic

Propositional vs. First-Order Inference

Unification and Lifting

Forward Chaining

Backward Chaining

Resolution

Natural number theory

- We present the theory of **natural numbers**
- We need
  - a predicate *NatNum* that will be true of natural numbers
  - one constant symbol, 0
  - one function symbol, S (successor)
  - one addition function, $+$
- The **Peano axioms** define natural numbers and addition. Natural numbers are defined recursively
  1. $NatNum(0)$
  2. $\forall n\, NatNum(n) \implies NatNum(S(n))$
  3. $\forall n\, 0 \neq S(n)$
  4. $\forall m, n\; m \neq n \implies S(m) \neq S(n)$
  5. $\forall m\, 0 \neq NatNum(m) \implies +(0, m) = m$
  6. $\forall m, n\; NatNum(m) \wedge NatNum(n) \implies +(S(m), n) = S(+(m, n))$

Set theory

- The domain of **sets** is also fundamental to mathematics as well as to commonsense reasoning
- We need
  - The empty set is a constant written as $\emptyset$
  - The unary predicate, Set, which is true of sets.
  - The infix binary predicate $x \in s$ (x is a member of set s)
  - The infix binary predicate $s_1 \subseteq s_2$ (set s_1 is a subset of set s_2)
  - The infix binary function $s_1 \cap s_2$ (the intersection of two sets)
  - The infix binary function $s_1 \cup s_2$ (the union of two sets)
  - The binary function $\{x \mid s\}$ (the set resulting from adjoining element x to set s)

Set theory (cont.)

One possible **set of axioms** is as follows

1. The only sets are the empty set and those made by adjoining something to a set
$$\forall s\, Set(s) \iff (s = \emptyset) \vee (\exists x, s_2\, Set(s_2) \wedge s = \{x \mid s_2\})$$
2. The empty set has no elements adjoined into it. In other words, there is no way to decompose $\emptyset$ into a smaller set and an element
$$\neg \exists x, s\ \{x \mid s\} = \emptyset.$$
3. Adjoining an element already in the set has no effect
$$\forall x, s\, x \in s \iff s = \{x \mid s\}.$$

Set theory (cont.)

4. The only members of a set are the elements that were adjoined into it. We express this recursively, saying that x is a member of s if and only if s is equal to some set s_2 adjoined with some element y, where either y is the same as x or x is a member of s_2

$$\forall x, s\; x \in s \iff \exists y, s_2\, (s = \{y \mid s_2\} \land (x = y \lor x \in s_2))\,.$$

5. A set is a subset of another set if and only if all of the first set's members are members of the second set

$$\forall s_1, s_2\; s_1 \subseteq s_2 \iff (\forall x\, x \in s_1 \Rightarrow x \in s_2).$$

6. Two sets are equal if and only if each is a subset of the other

$$\forall s_1, s_2\; s_1 = s_2 \iff (s_1 \subseteq s_2 \land s_2 \subseteq s_1).$$

Set theory (cont.)

7. An object is in the intersection of two sets if and only if it is a member of both sets

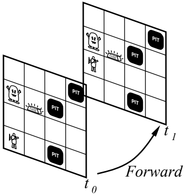
$$\forall x, s_1, s_2\ x \in (s_1 \cap s_2) \iff (x \in s_1 \land x \in s_2).$$

8. An object is in the union of two sets if and only if it is a member of either set

$$\forall x, s_1, s_2\ x \in (s_1 \cup s_2) \iff (x \in s_1 \lor x \in s_2).$$

Knowledge base for the wumpus world

- The corresponding first-order sentence stored in the knowledge base must include both the *percept* and the time t at which it occurred
- The actions in the wumpus world are also represented by logical terms



Agent

- **Perception**:
  $Percept([s, b, g, m, c], t), Stench(t), Breeze(t), Glitter(t)$
  $\textsc{Tell}(KB, \forall t, s, g, m, c\ Percept\ ([s, Breeze, g, m, c], t) \implies Breeze(t))$
  $\textsc{Tell}(KB, \forall t, s, b, m, c\ Percept\ ([s, b, Glitter, m, c], t) \implies Glitter(t))$
  $\textsc{Tell}(KB, Percept([Stench, Breeze, Glitter, None, None], 5))$

Knowledge base for the wumpus world (cont.)

- **Action**:
  $TurnRight$, $TurnLeft$, $Forward$, $Shoot$, $Grab$, $Climb$, $BestAction$
  For simple "reflex" behavior
  $\textsc{Tell}(KB, \forall t\ Glitter(t) \implies BestAction(Grab, t))$
  To determine which is best, the agent program executes the query
  $\textsc{AskVars}(KB, \exists a\ BestAction(a, t))$

Environment

$\textsc{Tell}(KB, \forall x, y, a, b\ Adjacent([x, y], [a, b]) \iff$
$(x = a \wedge (y = b - 1 \vee y = b + 1)) \vee (y = b \wedge (x = a - 1 \vee x = a + 1)))$

$\textsc{Tell}(KB, \forall x, s_1, s_2, t\ At(x, s_1, t) \wedge At(x, s_2, t) \implies s_1 = s_2)$

$\textsc{Tell}(KB, \forall s, t\ At(Agent, s, t) \wedge Breeze(t) \implies Breezy(s))$

$\textsc{Tell}(KB, \forall s\ Breezy(s) \iff \exists r\ Adjacent(r, s) \wedge Pit(r))$

Propositional vs. First-Order Inference

A brief history of reasoning

450B.C.	Stoics	propositional logic, inference (maybe)
322B.C.	Aristotle	"syllogisms" (inference rules), quantifiers
1565	Cardano	probability theory (propositional logic + uncertainty)
1847	Boole	propositional logic (again)
1879	Frege	first-order logic
1922	Wittgenstein	proof by truth tables
1930	Gödel	$\exists$ complete algorithm for FOL
1930	Herbrand	complete algorithm for FOL (reduce to propositional)
1931	Gödel	$\neg\exists$ complete algorithm for arithmetic
1960	Davis/Putnam	"practical" algorithm for propositional logic
1965	Robinson	"practical" algorithm for FOL – resolution

Universal instantiation (UI)

Concept 4

Every instantiation of a universally quantified sentence is entailed by it

$$\frac{\forall v\ \alpha}{\textsc{Subst}(\{v/g\}, \alpha)}$$

for any variable v and **ground term** g (a term without variables)

Example 1

$$\begin{aligned}
&\forall x\ King(x) \land Greedy(x) \implies Evil(x) \models \\
&\quad King(John) \land Greedy(John) \implies Evil(John) \\
&\quad King(Richard) \land Greedy(Richard) \implies Evil(Richard) \\
&\quad King(Father(John)) \land Greedy(Father(John)) \implies Evil(Father(John)) \\
&\quad \vdots
\end{aligned}$$

Representation Revisited

Syntax and Semantics of First-Order Logic (FOL)

Using First-Order Logic

Propositional vs. First-Order Inference

Unification and Lifting

Forward Chaining

Backward Chaining

Resolution

Existential instantiation (EI)

Concept 5

For any sentence α, variable v, and constant symbol k (**skolem constant**) *that does not appear elsewhere in the knowledge base*

$$\frac{\exists v\ \alpha}{\textsc{Subst}(\{v/k\}, \alpha)}$$

Example 2

$\exists x\ Crown(x) \land OnHead(x, John) \models$

$$Crown(C_1) \land OnHead(C_1, John)$$

provided C_1 is a new constant symbol

Representation Revisited
Syntax and Semantics of First-Order Logic (FOL)
Using First-Order Logic
Propositional vs. First-Order Inference
Unification and Lifting
Forward Chaining
Backward Chaining
Resolution

UI vs. EI

- UI can be applied several times to *add* new sentences; the new KB is logically equivalent to the old
- EI can be applied once to *replace* the existential sentence; the new KB is *not* equivalent to the old, but it can be shown to be **inferentially equivalent** (the new KB is satisfiable iff the old KB was satisfiable)

Reduction to propositional inference

- Suppose knowledge base KB contains just the sentences

$$\begin{aligned}
&\forall x\; King(x) \land Greedy(x) \implies Evil(x)\\
&King(John)\\
&Greedy(John)\\
&Brother(Richard, John).
\end{aligned}$$

- Instantiating the universal sentence in *all possible* ways, we have new KB

$$\begin{aligned}
&King(John) \land Greedy(John) \implies Evil(John)\\
&King(Richard) \land Greedy(Richard) \implies Evil(Richard)\\
&King(John)\\
&Greedy(John)\\
&Brother(Richard, John)
\end{aligned}$$

- The new KB is **propositionalized**: proposition symbols are

$$King(John), Greedy(John), Evil(John), King(Richard) \ldots$$

Representation Revisited
Syntax and Semantics of First-Order Logic (FOL)
Using First-Order Logic
Propositional vs. First-Order Inference
Unification and Lifting
Forward Chaining
Backward Chaining
Resolution

Reduction to propositional inference (cont.)

- **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., $Father(Father(Father(John)))$

Reduction to propositional inference (cont.)

Theorem 1 (Herbrand (1930))

If a sentence α is entailed by an FOL KB, it is entailed by a finite *subset of the propositional KB*

- **Idea:**

```
for n = 0 to ∞ do
    create a propositional KB by instantiating with
    depth-n terms see if α is entailed by this KB
```

- **Problem**: works if α is entailed, loops if α is not entailed

Theorem 2 (Turing (1936), Church (1936))

Entailment in FOL is ***semidecidable***

Unification and Lifting

Problems with propositionalization

- Propositionalization seems to generate lots of **irrelevant** sentences.
- For example, from

$$\begin{aligned}
&\forall x\, King(x) \land Greedy(x) \implies Evil(x)\\
&King(John)\\
&\forall y\, Greedy(y)\\
&Brother(Richard, John)
\end{aligned}$$

  it seems obvious that $Evil(John)$, but propositionalization produces lots of facts such as $Greedy(Richard)$ that are irrelevant
- With p k-ary predicates and n constants, there are $p \cdot n^k$ instantiations
- With function symbols, it gets nuch much worse!

Generalized Modus Ponens (GMP)

Generalized Modus Ponens

For atomic sentences p_i, p_i', and q, where there is a substitution θ such that $\textsc{Subst}(\theta, p_i') = \textsc{Subst}(\theta, p_i)$, for all i, (also write $\textsc{Subst}(\theta, p)$ as $p\theta$)

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{\textsc{Subst}(\theta, q)}$$

Example 3

For our example

p_1' is $King(John)$ | p_1 is $King(x)$
p_2' is $Greedy(y)$ | p_2 is $Greedy(x)$
θ is $\{x/John, y/John\}$ | q is $Evil(x)$
$\textsc{Subst}(\theta, q)$ is $Evil(John)$.

Soundness of GMP

Lemma 1

(self exercise) For any definite clause p, we have $p \models p\theta$ by UI

Proof

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

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. $p_1', \ \ldots, \ p_n' \models p_1' \land \ldots \land p_n' \models p_1'\theta \land \ldots \land p_n'\theta$
3. From 1 and 2, $q\theta$ follows by ordinary Modus Ponens

■

Unification

Concept 6

Unification is a process to find substitutions θ that make different logical expressions p and q look identical.

$$\text{UNIFY}(p, q) = \theta \text{ where } \text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)$$

Example 4

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John, x/Mother(John)\}$
$Knows(John, x)$	$Knows(x, OJ)$	$fail$

Representation Revisited
Syntax and Semantics of First-Order Logic (FOL)
Using First-Order Logic
Propositional vs. First-Order Inference
Unification and Lifting
Forward Chaining
Backward Chaining
Resolution

Most General Unifier (MGU)

- Consider the unification $\textsc{Unify}(Knows(John, x), Knows(y, z))$, the results could be
  - $\theta_1 = \{y/John, x/z\}$
  - $\theta_2 = \{y/John, x/John, z/John\}$
- 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

$$\theta_{MGU} = \{y/John, x/z\}$$

The unification algorithm

```
function UNIFY(x, y, θ) returns a substitution to make x and y identical
inputs: x, a variable, constant, list, or compound
        y, a variable, constant, list, or compound
        θ, the substitution built up so far (optional, defaults to empty)
  if θ = failure then return failure
  else if x = y then return θ
  else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
  else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
  else if COMPOUND?(x) and COMPOUND?(y) then return UNIFY(x.ARGS, y.ARGS, UNIFY(x.OP, y.OP, θ))
  else if LIST?(x) and LIST?(y) then return UNIFY(x.REST, y.REST, UNIFY(x.FIRST, y.FIRST, θ))
  else return failure

function UNIFY-VAR(var, x, θ) returns a substitution
  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
```

Forward Chaining

First-order definite clauses

- **A definite clause** is a disjunctions of literals of which exactly one is positive. It is
  - an atomic or
  - an implication whose antecedent is a conjunctions of positive literals and consequent is a positive literal

  $King(x) \wedge Greedy(x) \Rightarrow Evil(x).$
  $King(John).$
  $Greedy(y).$
- **A first-order literal** can include variables, which are assumed to be universally quantified
- Not every knowledge base can be converted into a set of definite clauses because of the **single-positive-literal** restriction

Example knowledge base

Problem

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** Colonel West is a criminal?

- ... it is a crime for an American to sell weapons to hostile nations

$$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \implies Criminal(x)$$

- Nono ... has some missiles

$$\exists\, x Owns(Nono, x) \land Missile(x)$$

Example knowledge base (cont.)

$$Owns(Nono, M_1) \text{ and } Missile(M_1) \text{ (EI)}$$

- ... all of its missiles were sold to it by Colonel West

$$\forall x\, Missile(x) \land Owns(Nono, x) \implies Sells(West, x, Nono)$$

- Missiles are weapons

$$Missile(x) \implies Weapon(x)$$

- An enemy of America counts as "hostile"

$$Enemy(x, America) \implies Hostile(x)$$

- West, who is American ...

$$American(West)$$

Representation Revisited

Syntax and Semantics of First-Order Logic (FOL)

Using First-Order Logic

Propositional vs. First-Order Inference

Unification and Lifting

Forward Chaining

Backward Chaining

Resolution

Example knowledge base (cont.)

- The country Nono, an enemy of America ...

$$Enemy(Nono, America)$$

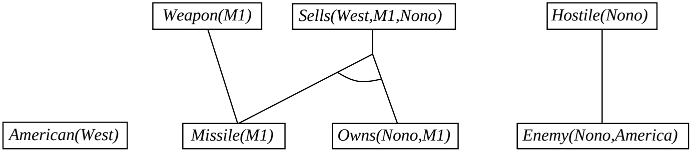
Forward chaining algorithm

```
function FOL-FC-ASK(KB, α) returns a substitution or false
inputs: KB, the knowledge base, a set of first order definite clauses
        α, the query, an atomic sentence
local variables: new, the new sentences inferred on each iteration
  repeat until new = ∅
    new ← ∅
    for each rule in KB do
    (p1 ∧ ... ∧ pn ⟹ q) ← STANDARDIZE-VARIABLES(rule)
    for each θ such that SUBST(θ, p1 ∧ ... ∧ pn) = SUBST(θ, p'1 ∧ ... ∧ p'n)
                for some p'1 ∧ ... ∧ p'n in KB
      q' ← SUBST(θ, q)
      if q' does not unify with some sentence already in KB or new then
        add q' to new
        φ ← UNIFY(q', α)
        if φ is not fail then return φ
    add new to KB
  return false
```

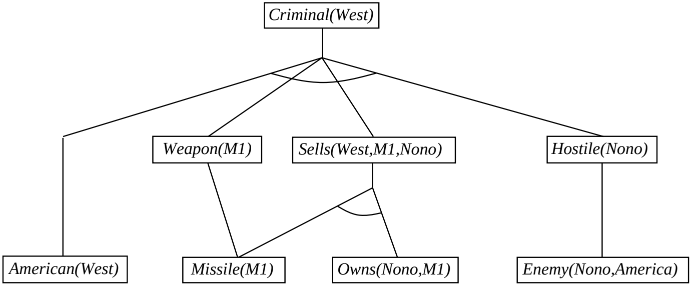
Forward chaining proof

American(West) *Missile(M1)* *Owns(Nono,M1)* *Enemy(Nono,America)*

Forward chaining proof (cont.)



Forward chaining proof (cont.)



Properties of forward chaining

- **Sound**:
  - YES, every inference is just an application of GMP
- **Complete**:
  - YES for definite clause knowledge bases
  - It answers every query whose answers are entailed by any *KB* of definite clauses
- **Datalog** = first-order definite clauses + *no functions* (e.g., crime KB)
- FC terminates for Datalog in poly iterations: at most $p \cdot n^k$ literals
- May not terminate in general if α is not entailed
  - This is unavoidable: entailment with definite clauses is **semidecidable**

Efficiency of forward chaining

- **Simple observation**: no need to match a rule on iteration k if a premise wasn't added on iteration $k-1$
  $\rightarrow$ match each rule whose premise contains a newly added literal
- Matching itself can be expensive
- **Database indexing** allows $O(1)$ retrieval of known facts
  E.g., query $\mathit{Missile}(x)$ retrieves $\mathit{Missile}(M_1)$
- Matching conjunctive premises against known facts is NP-hard
- Forward chaining is widely used in **deductive databases**

Backward Chaining

A backward-chaining algorithm

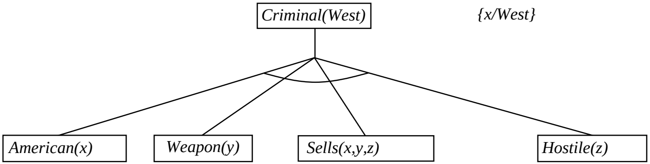
```
function FOL-BC-ASK(KB, query) returns a generator of substitutions
  return FOL-BC-OR(KB, query, ∅)

generator FOL-BC-OR(KB, goal, θ) yields a substitution
  for each rule (lhs ⟹ rhs) in FETCH-RULES-FOR-GOAL(KB, goal) do
    (lhs, rhs) ← STANDARDIZE-VARIABLES((lhs, rhs))
    for each θ′ in FOL-BC-AND(KB, lhs, UNIFY(rhs, goal, θ)) do
      yield θ′

generator FOL-BC-AND(KB, goals, θ) yields a substitution
  if θ = failure then return
  else if length(goals) = 0 then yield θ
  else do
    first, rest ← FIRST(goals), REST(goals)
    for each θ′ in FOL-BC-OR(KB, SUBST(θ, first), θ) do
      for each θ″ in FOL-BC-AND(KB, rest, θ′) do
        yield θ″
```

Representation Revisited
Syntax and Semantics of First-Order Logic (FOL)
Using First-Order Logic
Propositional vs. First-Order Inference
Unification and Lifting
Forward Chaining
Backward Chaining
Resolution

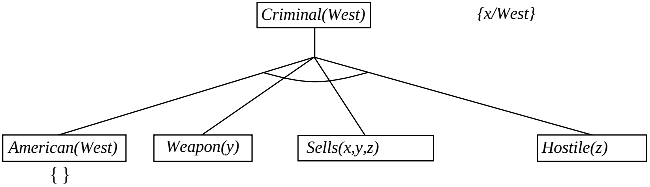
Backward chaining example

$\boxed{Criminal(West)}$

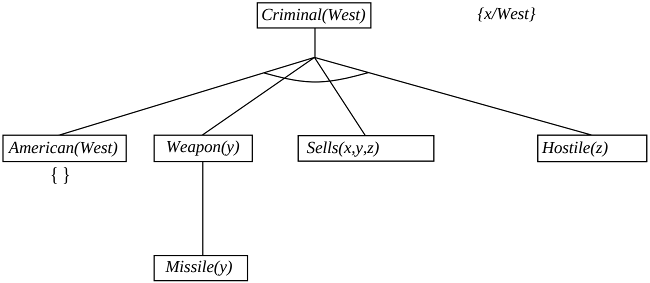
Backward chaining example (cont.)



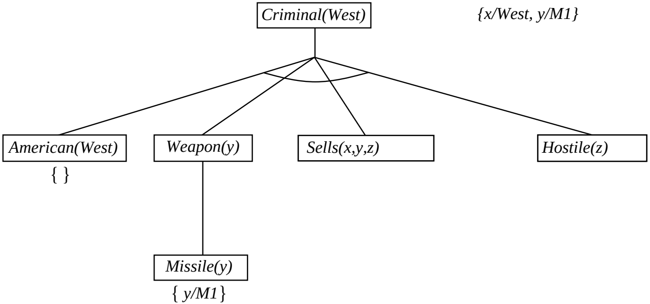
Backward chaining example (cont.)



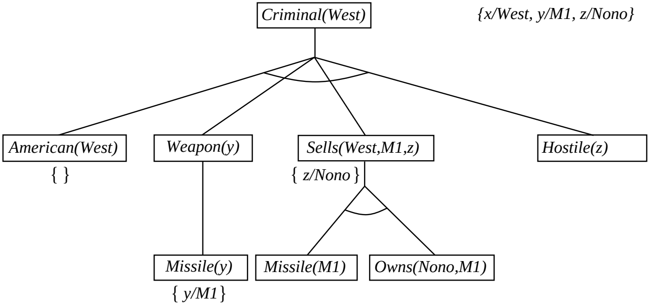
Backward chaining example (cont.)



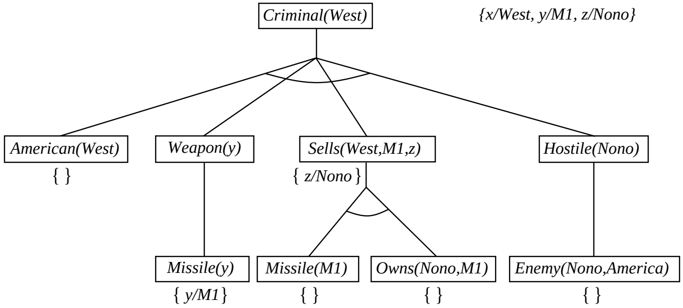
Backward chaining example (cont.)



Backward chaining example (cont.)



Backward chaining example (cont.)



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**

Resolution

Resolution: brief summary

Concept 7

Full first-order version

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \; m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

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

Example 5

$$\frac{\neg Rich(x) \vee Unhappy(x), \; Rich(Ken)}{Unhappy(Ken)}$$

with $\theta = \{x/Ken\}$

- Apply resolution steps to $CNF(KB \wedge \neg\alpha)$; complete for FOL

Conversion to CNF

A sentence "Everyone who loves all animals is loved by someone" is represented by

$$\forall x [\forall y \; Animal(y) \implies Loves(x,y)] \implies [\exists y \; Loves(y,x)]$$

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: $\neg \forall x, p \equiv \exists x \neg p$, $\neg \exists x, p \equiv \forall x \neg p$

$$\forall x [\exists y \; \neg(\neg Animal(y) \lor Loves(x,y))] \lor [\exists y \; Loves(y,x)]$$
$$\forall x [\exists y \; \neg\neg Animal(y) \land \neg Loves(x,y)] \lor [\exists y \; Loves(y,x)]$$
$$\forall x [\exists y \; Animal(y) \land \neg Loves(x,y)] \lor [\exists y \; Loves(y,x)]$$

Conversion to CNF (cont.)

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

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

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

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

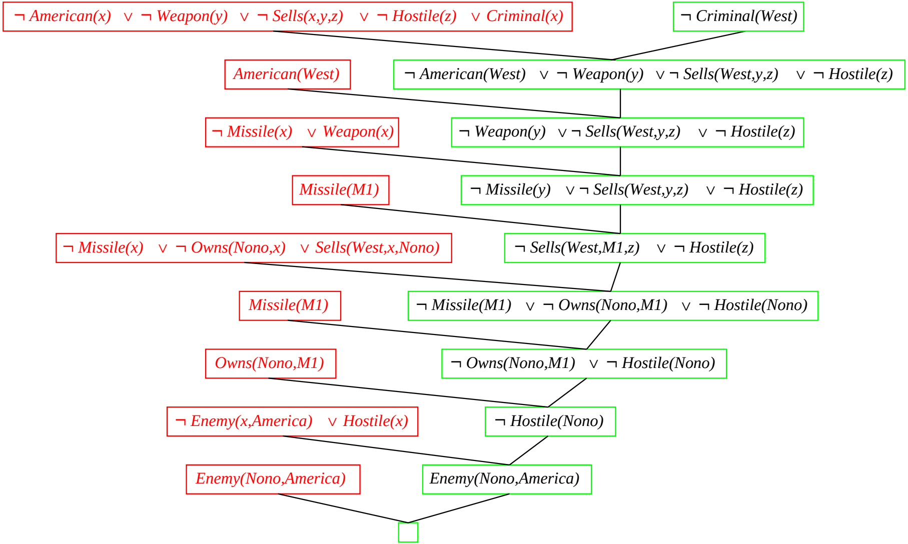
5. Drop universal quantifiers

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

6. Distribute $\land$ over $\lor$

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

Resolution proof: definite clauses



References

Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep learning.
MIT press.

Lê, B. and Tô, V. (2014).
Cở sở trí tuệ nhân tạo.
Nhà xuất bản Khoa học và Kỹ thuật.

Nguyen, T. (2018).
Artificial intelligence slides.
Technical report, HCMC University of Sciences.

Russell, S. and Norvig, P. (2016).
Artificial intelligence: a modern approach.
Pearson Education Limited.