First-order Logic

Bùi Tiến Lên

2024

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

Contents

1. Representation Revisited

2. Syntax and Semantics

3. Applications

4. Simple Inference

5. Unification

6. Forward Chaining

7. Backward Chaining

8. Resolution

Representation Revisited

Programming languages

Concept 1

Programming language is a kind of formal languages.

- **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.
- **Predicates**: are relations that return true/false
- **Functions**: are relations that return object

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

BNF Grammar

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

Representation Revisited
Syntax and Semantics
Applications
Simple Inference
Unification
Forward Chaining
Backward Chaining
Resolution

Terms

Concept 2

Term is a logical expression that refers to an object

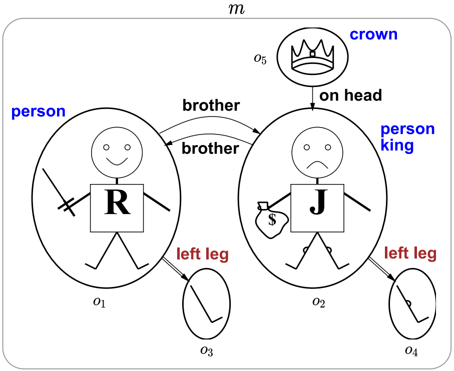
- Constants
- Functions
- Variables

Concept 3

Ground term is a term without variables

Graph representation of a model

- A model can be represented as a directed graph.
- The following graph contains five **objects**, two **binary relations**, three **unary relations** (indicated by labels on the objects), and one **unary function**, left-leg.



Models in First-order logic

Concept 4

A model m in first-order logic maps:

- Constant symbols to objects

$$m(R) = o_1 \text{ and } m(J) = o_2$$

- Predicate/function symbols to tuples of objects

$$\begin{aligned} m(Person) &= \{o_1, o_2\} \\ m(King) &= \{o_2\} \\ m(Crown) &= \{o_5\} \\ m(Brother) &= \{(o_1, o_2), (o_2, o_1)\} \\ m(onhead) &= \{(o_5, o_2)\} \\ m(leftleg) &= \{(o_1, o_3), (o_2, o_4)\} \end{aligned}$$

Models in First-order logic (cont.)

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

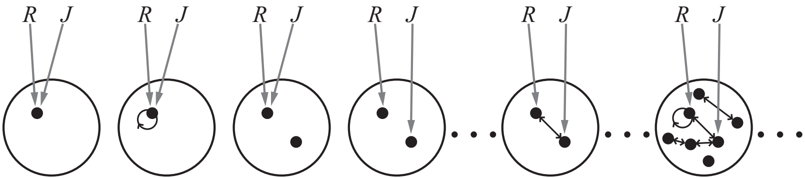
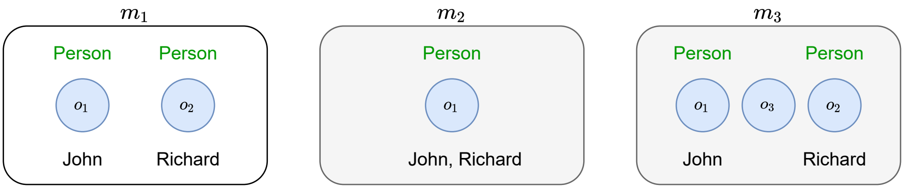


Figure 1: 137,506,194,466 models with six or fewer objects.

A restriction on models

"Richard and John are people."

$$Person(Richard) \land Person(John)$$



- **Unique names assumption**: Each object has **at most** one constant symbol. This rules out m_2.
- **Domain closure**: Each object has **at least** one constant symbol. This rules out m_3.

Point:

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

Quantifiers

Concept 5 (Universal quantification)

$\forall \langle variables \rangle \ \langle sentence \rangle$ $\forall x \ P(x)$ is true in a model m iff $P(x)$ is true with x being *each* possible object in the model

- Think conjunction: $\forall x \ P(x)$ is like $P(A) \wedge P(B) \wedge \ldots$

Concept 6 (Existential quantification)

$\exists \langle variables \rangle \ \langle sentence \rangle$ $\exists x \ P(x)$ is true in a model m iff $P(x)$ is true with x being *some* possible object in the model

- Think disjunction: $\exists x \ P(x)$ is like $P(A) \vee P(B) \vee \ldots$

Natural language quantifiers

- "Everyone at Berkeley is smart"

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

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

$$\begin{array}{ll} & (At(King\ John, Berkeley) \implies Smart(King\ John)) \\ \wedge & (At(Richard, Berkeley) \implies Smart(Richard)) \\ \wedge & (At(Berkeley, Berkeley) \implies Smart(Berkeley)) \\ \wedge & \ldots \end{array}$$

Natural language quantifiers (cont.)

- "Someone at Stanford is smart"

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

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

$$\begin{aligned} & (At(King\ John, Berkeley) \wedge Smart(King\ John)) \\ \vee \quad & (At(Richard, Berkeley) \wedge Smart(Richard)) \\ \vee \quad & (At(Berkeley, Berkeley) \wedge Smart(Berkeley)) \\ \vee \quad & \ldots \end{aligned}$$

A common mistake to avoid

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

Nested quantifiers

Multiple quantifiers enable more complex sentences. **The order** of quantification is therefore very important.

- Simplest cases: Quantifiers are of the same type
  $\forall x \forall y \ (Brother(x, y) \implies Sibling(x, y))$
  $\forall x \forall y \ (Sibling(x, y) \iff Sibling(y, x))$
- 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"

Nested quantifiers (cont.)

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

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

Rule:

- The variable belongs to the **innermost** quantifier that mentions it or
- Use different variable names with nested quantifier

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

Properties of quantifiers

- Nested quantifiers

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

- De Morgan's rules

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

Equality

Concept 7

- $term_1 = term_2$ 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

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

$$\forall x, y \ (Mother(x, y) \iff (Female(x) \land Parent(x, y)))$$

- A first cousin is a child of a parent's sibling

$$\forall x, y \ (FirstCousin(x, y) \iff \exists p, ps \ (Parent(p, x) \land Sibling(ps, p) \land Parent(ps, y)))$$

Applications

Using First-Order Logic

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 $\textsc{Ask}$

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

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

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

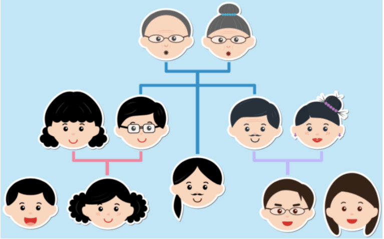
Using First-Order Logic (cont.)

- 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

$$\forall g, c\,(GrandParent(g, c) \iff \exists p\; Parent(g, p) \land Parent(p, c)).$$

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

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

Using axioms to entail theorems

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

Natural number theory

- To 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
$$\forall s \; (Set(s) \iff (s = \emptyset) \lor (\exists x, s_2 \; Set(s_2) \land s = \{x \mid s_2\}))$$
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

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

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

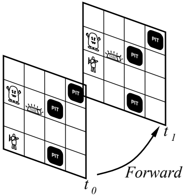
$$\forall s_1, s_2 \; (s_1 = s_2 \iff (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)) \, .$$

Set theory (cont.)

7. An object is in the intersection of two sets if and only if it is a member of both sets
$$\forall x, s_1, s_2 \ (x \in (s_1 \cap s_2) \iff (x \in s_1 \wedge x \in s_2)) .$$
8. An object is in the union of two sets if and only if it is a member of either set
$$\forall x, s_1, s_2 \ (x \in (s_1 \cup s_2) \iff (x \in s_1 \vee x \in s_2)) .$$

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

$$\textsc{Tell}(KB, \forall x, y, a, b\ (Adjacent([x, y], [a, b]) \iff$$
$$(x = a \wedge (y = b - 1 \vee y = b + 1)) \vee (y = b \wedge (x = a - 1 \vee x = a + 1))))$$
$$\textsc{Tell}(KB, \forall x, s_1, s_2, t\ (At(x, s_1, t) \wedge At(x, s_2, t) \implies s_1 = s_2))$$
$$\textsc{Tell}(KB, \forall s, t\ (At(Agent, s, t) \wedge Breeze(t) \implies Breezy(s)))$$
$$\textsc{Tell}(KB, \forall s\ (Breezy(s) \iff \exists r\ Adjacent(r, s) \wedge Pit(r)))$$

Simple 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

Substitution

Concept 8

A substitution θ is a mapping from variables to terms. $\textsc{Subst}[\theta, \alpha]$ returns the result of performing substitution θ on α. Note that: $\textsc{Subst}[\theta, \alpha]$ also writtened as $\alpha\theta$.

- $\textsc{Subst}(\{x/alice\}, P(x)) = P(alice)$
- $\textsc{Subst}(\{x/alice, y/z\}, P(x) \wedge Q(x, y) = P(alice) \wedge Q(alice, z)$

Skolem normal form

Concept 9

A sentence of first-order logic is in **Skolem normal form** if it is written as a string of quantifiers and variables (with only universal first-order quantifiers) followed by a quantifier-free part.

- Every first-order sentence may be converted into Skolem normal form while not changing its satisfiability.

Universal instantiation (UI)

Concept 10

Every instantiation of a universally quantified sentence is entailed by it

$$\frac{\forall x \; \alpha}{\textsc{Subst}(\{x/g\}, \alpha)} \tag{1}$$

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

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

Existential instantiation (EI)

Concept 11

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

$$\frac{\exists x\ \alpha}{\textsc{Subst}(\{x/k\}, \alpha)} \tag{2}$$

$$\exists x\ (Crown(x) \land OnHead(x, John)) \models Crown(C_1) \land OnHead(C_1, John)$$

provided C_1 is a new constant symbol

Existential instantiation (EI) (cont.)

- The logic equivalence

$$\forall x \, \exists y \, R(x, y) \iff \exists y \, \forall x \, R(x, f(x)) \tag{3}$$

where $f(x)$ is a function that maps x to y.

Concept 12

For any sentence α, variable x, y, and and function f (**skolem function**)

$$\frac{\forall x \, \exists y \, \alpha}{\textsc{Subst}(\{y/f(x)\}, \alpha)} \tag{4}$$

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)

Propositionalization

- Knowledge base KB in first-order logic

$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John).$$
$$\forall x \; (King(x) \land Greedy(x) \implies Evil(x))$$

$\downarrow$ (Instantiating)

- Knowledge base in propositional logic

$$King\text{-}John$$
$$Greedy\text{-}John$$
$$Brother\text{-}Richard\text{-}John$$
$$King\text{-}John \land Greedy\text{-}John \implies Evil\text{-}John$$
$$King\text{-}Richard \land Greedy\text{-}Richard \implies Evil\text{-}Richard$$

Propositionalization (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., $father(father(father(John)))$

Propositionalization (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

Propositionalization (cont.)

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

Entailment in FOL is ***semidecidable***

- Algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every non-entailed sentence.

Problems with propositionalization

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

$$King(John)$$
$$Brother(Richard, John)$$
$$\forall x \; Greedy(x)$$
$$\forall x \; (King(x) \wedge Greedy(x) \implies Evil(x))$$

  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!

Unification

Unification

Concept 13

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

$$\textsc{Unify}(p, q) = \theta \text{ where } \textsc{Subst}(\theta, p) = \textsc{Subst}(\theta, q)$$

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

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 θ_1 is more general than the second θ_2
- 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(s1, s2, θ) returns a substitution to make s1 and s2 identical
inputs: s1, a variable, constant, list, or compound
        s2, 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 s1 = s2 then return θ
  else if VARIABLE?(s1) then return UNIFY-VAR(s1, s2, θ)
  else if VARIABLE?(s2) then return UNIFY-VAR(s2, s1, θ)
  else if COMPOUND?(s1) and COMPOUND?(s2) then return UNIFY(s1.ARGS, s2.ARGS, UNIFY(s1.OP, s2.OP, θ))
  else if LIST?(s1) and LIST?(s2) then return UNIFY(s1.REST, s2.REST, UNIFY(s1.FIRST, s2.FIRST, θ))
  else return failure

function UNIFY-VAR(var, s, θ) returns a substitution
  if {var/val} ∈ θ then return UNIFY(val, s, θ)
  else if {s/val} ∈ θ then return UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, s) then return failure
  else return add {var/s} to θ
```

Occur Check

Given $\textsc{Unify-Var}(var, s)$ return failure if where var occurs in s and s is not a variable

- For example, $\textsc{Unify-Var}(x, father(x))$ cannot be unified.

Quiz: Unification

p	q	θ
$P(f(A), g(x))$	$P(y, y)$	
$P(A, x, h(g(z)))$	$P(z, h(y), h(y))$	
$P(x, f(x), z)$	$P(g(y), f(g(B)), y)$	
$P(x, f(x))$	$P(f(y), y)$	
$P(x, f(z))$	$P(f(y), y)$	

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
- **A first-order literal** can include variables, which are assumed to be *universally quantified*

$$King(x) \land Greedy(x) \Rightarrow Evil(x)$$
$$King(John)$$
$$Greedy(y)$$

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

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

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{\textsc{Subst}(\theta, q)} \tag{5}$$

For example

$$\begin{array}{ll} p_1' \text{ is } King(John) & p_1 \text{ is } King(x) \\ p_2' \text{ is } Greedy(y) & p_2 \text{ is } Greedy(x) \\ & q \text{ is } Evil(x) \\ \hline \theta \text{ is } \{x/John, y/John\} & \textsc{Subst}(\theta, q) \text{ is } Evil\text{-}John \end{array}$$

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

$$p'_1, \ldots, p'_n, (p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models q\theta$$

provided that $p'_i\theta = p_i\theta$ for all i

1. $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models (p_1 \wedge \ldots \wedge p_n \Rightarrow q)\theta = (p_1\theta \wedge \ldots \wedge p_n\theta \Rightarrow q\theta)$
2. $p'_1, \ldots, p'_n \models p'_1 \wedge \ldots \wedge p'_n \models p'_1\theta \wedge \ldots \wedge p'_n\theta$
3. From 1 and 2, $q\theta$ follows by ordinary Modus Ponens

■

Example 1

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?

Example 1 (cont.)

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

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

- Nono … has some missiles

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

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

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

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

- Missiles are weapons

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

Example 1 (cont.)

- An enemy of America counts as “hostile”

$$\mathit{Enemy}(x, \mathit{America}) \implies \mathit{Hostile}(x)$$

- West, who is American ...

$$\mathit{American}(\mathit{West})$$

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

$$\mathit{Enemy}(\mathit{Nono}, \mathit{America})$$

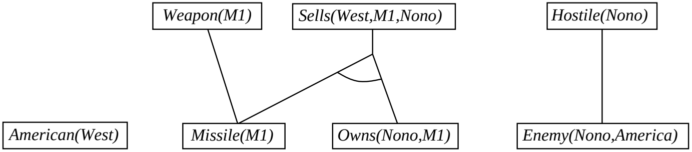
Forward chaining algorithm

```
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
```

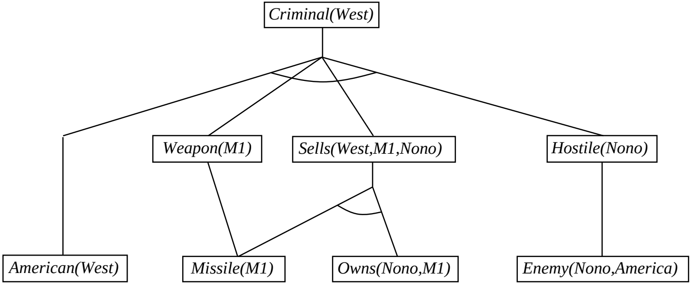
Forward chaining proof

$\boxed{\textit{American(West)}}$ $\boxed{\textit{Missile(M1)}}$ $\boxed{\textit{Owns(Nono,M1)}}$ $\boxed{\textit{Enemy(Nono,America)}}$

Forward chaining proof (cont.)



Forward chaining proof (cont.)



Example 2

Problem

- Art is the father of Bob and Bud.
- Bob is the father of Cal and Coe.
- Grandfather is the father of a father.

Is Art the grandfather of Coe?

Example 2 (cont.)

Convert English sentences into FOL sentences

#	FOL sentence	explain
1	$F(Art, Bob)$	KB
2	$F(Art, Bud)$	KB
3	$F(Bob, Cal)$	KB
4	$F(Bob, Coe)$	KB
5	$F(x, y) \wedge F(y, z) \implies G(x, z)$	KB
6	$G(Art, Coe)$	1,4,5 $\{x/Art, y/Bob, z/Coe\}$

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
- It terminates for Datalog in poly iterations: at most $p \cdot n^k$ literals
- It may not terminate in general if α is not entailed
  - This is unavoidable: entailment with definite clauses is **semidecidable**

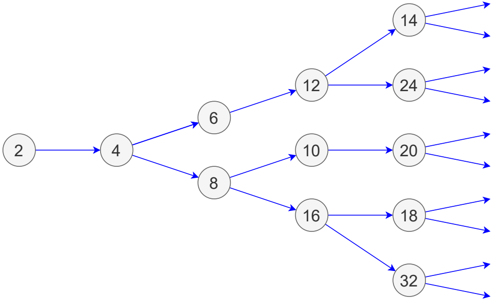
Efficiency of forward chaining

- **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**

Definite clauses with function symbols

- Inference can explode forward and may never terminate.
- Consider the following KB with two predicates and two functions

$$Even(2)$$
$$Even(x) \implies Even(plus(x, 2))$$
$$Integer(x) \implies Even(times(x, 2))$$
$$Even(x) \implies Integer(x)$$



Quiz: Forward chaining

- Given a KB containing the following sentence
  $Parent(x, y) \land Male(x) \implies Father(x, y)$
  $Father(x, y) \land Father(x, z) \implies Sibling(y, z)$
  $Parent(Tom, John)$
  $Male(Tom)$
  $Parent(Tom, Fred)$
- Perform the forward chaining until a fixed point is reached.

Backward Chaining

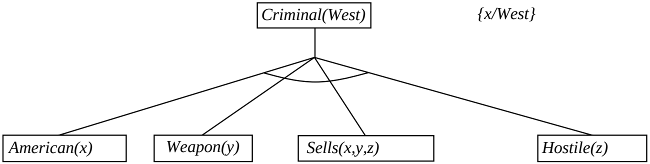
A backward-chaining algorithm

```
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″
```

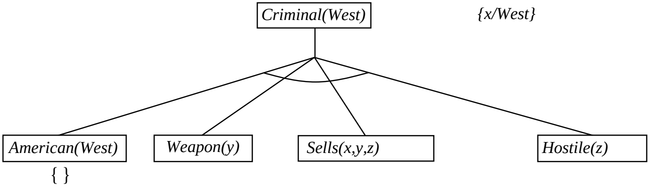
Backward chaining example

$\boxed{Criminal(West)}$

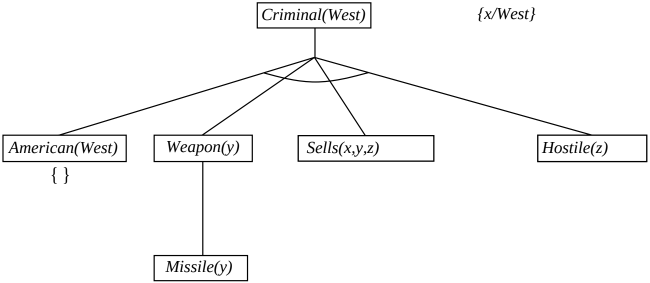
Backward chaining example (cont.)



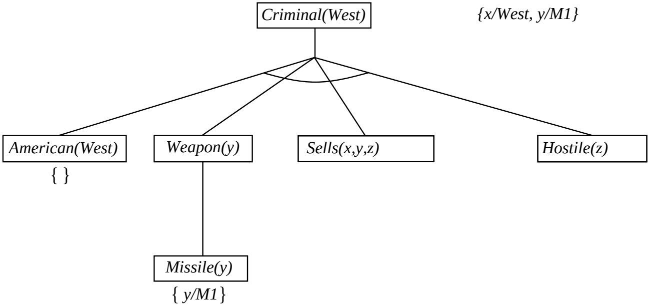
Backward chaining example (cont.)



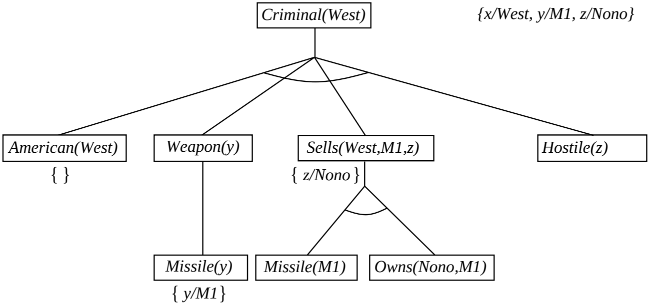
Backward chaining example (cont.)



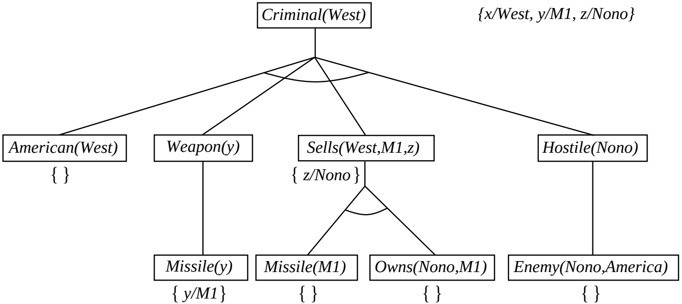
Backward chaining example (cont.)



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

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. Standardize variables: each quantifier should use a different one

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

2. Eliminate biconditionals and implications

$$\forall x [[\neg \forall y\ [\neg Animal(y) \vee Loves(x, y)]] \vee [\exists z\ Loves(z, x)]]$$

Conversion to CNF (cont.)

3. Move $\neg$ inwards:

$$\forall x[[\exists y\ \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists z\ Loves(z,x)]]$$
$$\forall x[[\exists y\ (\neg\neg Animal(y) \wedge \neg Loves(x,y))] \vee [\exists z\ Loves(z,x)]]$$
$$\forall x[[\exists y\ (Animal(y) \wedge \neg Loves(x,y))] \vee [\exists z\ Loves(z,x)]]$$

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

$$\forall x[[Animal(F(x)) \wedge \neg Loves(x,f(x))] \vee Loves(g(x),x)]$$

5. Drop universal quantifiers

$$[Animal(f(x)) \wedge \neg Loves(x,f(x))] \vee Loves(g(x),x)$$

Conversion to CNF (cont.)

6. Distribute $\wedge$ over $\vee$

$$[Animal(f(x)) \vee Loves(g(x), x)] \wedge [\neg Loves(x, f(x)) \vee Loves(g(x), x)]$$

Generalized Resolution

Concept 14

Full first-order version

$$\frac{\ell_1 \vee \cdots \vee \ell_i \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_j \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta} \tag{6}$$

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

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

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

Solution to Example 1

$\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x,y,z) \vee \neg Hostile(z) \vee \mathbf{Criminal(x)}$	$\neg \mathbf{Criminal(West)}$
$\mathbf{American(West)}$	$\neg \mathbf{American(West)} \vee \neg Weapon(y) \vee \neg Sells(West,y,z) \vee \neg Hostile(z)$
$\neg Missile(x) \vee \mathbf{Weapon(x)}$	$\neg \mathbf{Weapon(y)} \vee \neg Sells(West,y,z) \vee \neg Hostile(z)$
$\mathbf{Missile(M_1)}$	$\neg \mathbf{Missile(y)} \vee \neg Sells(West,y,z) \vee \neg Hostile(z)$
$\neg Missile(x) \vee \neg Owns(Nono,x) \vee \mathbf{Sells(West,x,Nono)}$	$\neg \mathbf{Sells(West,M_1,z)} \vee \neg Hostile(z)$
$\mathbf{Missile(M_1)}$	$\neg \mathbf{Missile(M_1)} \vee \neg Owns(Nono,M_1) \vee \neg Hostile(Nono)$
$\mathbf{Owns(Nono,M_1)}$	$\neg \mathbf{Owns(Nono,M_1)} \vee \neg Hostile(Nono)$
$\neg Enemy(x,America) \vee \mathbf{Hostile(x)}$	$\neg \mathbf{Hostile(Nono)}$
$\mathbf{Enemy(Nono,America)}$	$\neg \mathbf{Enemy(Nono,America)}$
	$\square$

Solution to Example 2

#	FOL clause	explain
1	$F(Art, Bob)$	KB
2	$F(Art, Bud)$	KB
3	$F(Bob, Cal)$	KB
4	$F(Bob, Coe)$	KB
5	$\neg F(x, y) \vee \neg F(y, z) \vee G(x, z)$	KB
6	$\neg G(Art, Coe)$	$\neg\alpha$
7	$\neg F(Art, y) \vee \neg F(y, Code)$	5,6 $\{x/Art, z/Coe\}$
8	$\neg F(Art, Bob)$	4,7 $\{y/Bob\}$
9	$\emptyset$	1,8

Example 3

Problem

Everyone who loves all animals is loved by someone. Anyone who kills an animal is loved by no one. Jack loves all animals. Either Jack or Curiosity killed the cat, who is named Tuna. Did Curiosity kill the cat?

Solution to Example 3

1. Everyone who loves animals is loved by someone.
2. Anyone who kills an animal is loved by no one.
3. Jack loves all animals.
4. Either Jack or Curiosity killed the cat.
5. The cat is named Tuna.
6. Cats are animals.
7. Did Curiosity kill the cat?

Solution to Example 3 (cont.)

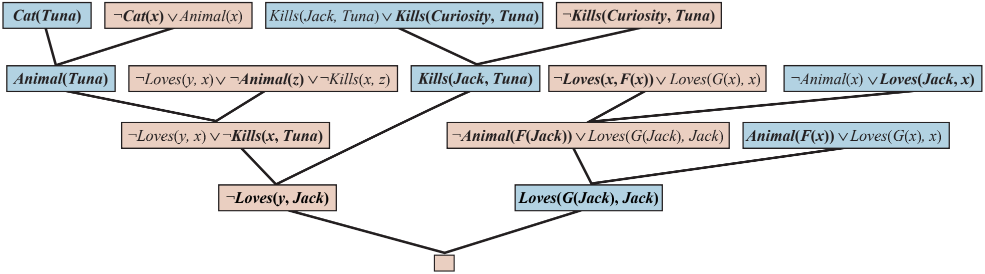
1. $\forall x[[\forall y Animal(y) \implies Loves(x,y)] \implies \exists z Loves(z,x)]$
2. $\forall x[[\exists y Animal(y) \land Kills(x,y)] \implies \forall z \neg Loves(z,x)]$
3. $\forall x[Animal(x) \implies Loves(Jack,x)]$
4. $Kills(Jack, Tuna) \lor Kills(Curiosity, Tuna)$
5. $Cat(Tuna)$
6. $\forall x[Cat(x) \implies Animal(x)]$
7. $\neg Kills(Curiosity, Tuna)$

Solution to Example 3 (cont.)

1. $Animal(F(x)) \lor Loves(G(x), x)$
2. $\neg Loves(x, F(x)) \lor Loves(G(x), x)$
3. $\neg Loves(y, x) \lor \neg Animal(z) \lor \neg Kills(x, z)$
4. $\neg Animal(x) \lor \neg Loves(Jack, x)$
5. $Kills(Jack, Tuna) \lor Kills(Curiousity, Tuna)$
6. $Cat(Tuna)$
7. $\neg Cat(x) \lor Animal(x)$
8. $\neg Kills(Curiousity, Tuna)$

Solution to Example 3 (cont.)

Suppose Curiosity did not kill Tuna. We know that either Jack or Curiosity did; thus Jack must have. Now, Tuna is a cat and cats are animals, so Tuna is an animal. Because anyone who kills an animal is loved by no one, we know that no one loves Jack. On the other hand, Jack loves all animals, so someone loves him; so we have a contradiction. Therefore, Curiosity killed the cat.



Exercise

Given a KB of the following sentences

- Anyone whom Mary loves is a football star.
- Any student who does not pass does not play.
- John is a student.
- Any student who does not study does not pass.
- Anyone who does not play is not a football star.

Prove that "If John does not study, Mary does not love John"

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. (2021).
Artificial intelligence: a modern approach.
Pearson Education Limited.