Introduction to Artificial Intelligence

Lecture: First Order Logic

Outline

- First-Order Logic (FOL)
- Unification and Lifting
- Forward Chaining
- Backward Chaining
- Resolution

First-order Logic

- Objects are referred by nouns and noun phrases.
  - E.g., people, houses, numbers, colors, Bill Gates, games, wars, etc.
- Relations can be unary relations (properties) or $\boldsymbol{n}$-ary relations, representing by verbs and verb phrases
  - Properties: red, round, prime, etc.
  - n-ary relations: brother of, bigger than, part of, comes between, etc.
- Functions are relations in which there is only one "value" for a given "input."
  - E.g., father of, best friend, one more than, etc.

First-order Logic

- “One plus two equals three.”
  - Object: one, two, three, one plus two
  - Relation: equal
  - Function: plus
- “The intelligent AlphaGo beat the world champion in 2016.”
  - Object: AlphaGo, world champion, 2016
  - Property: intelligent
  - Relation: beat

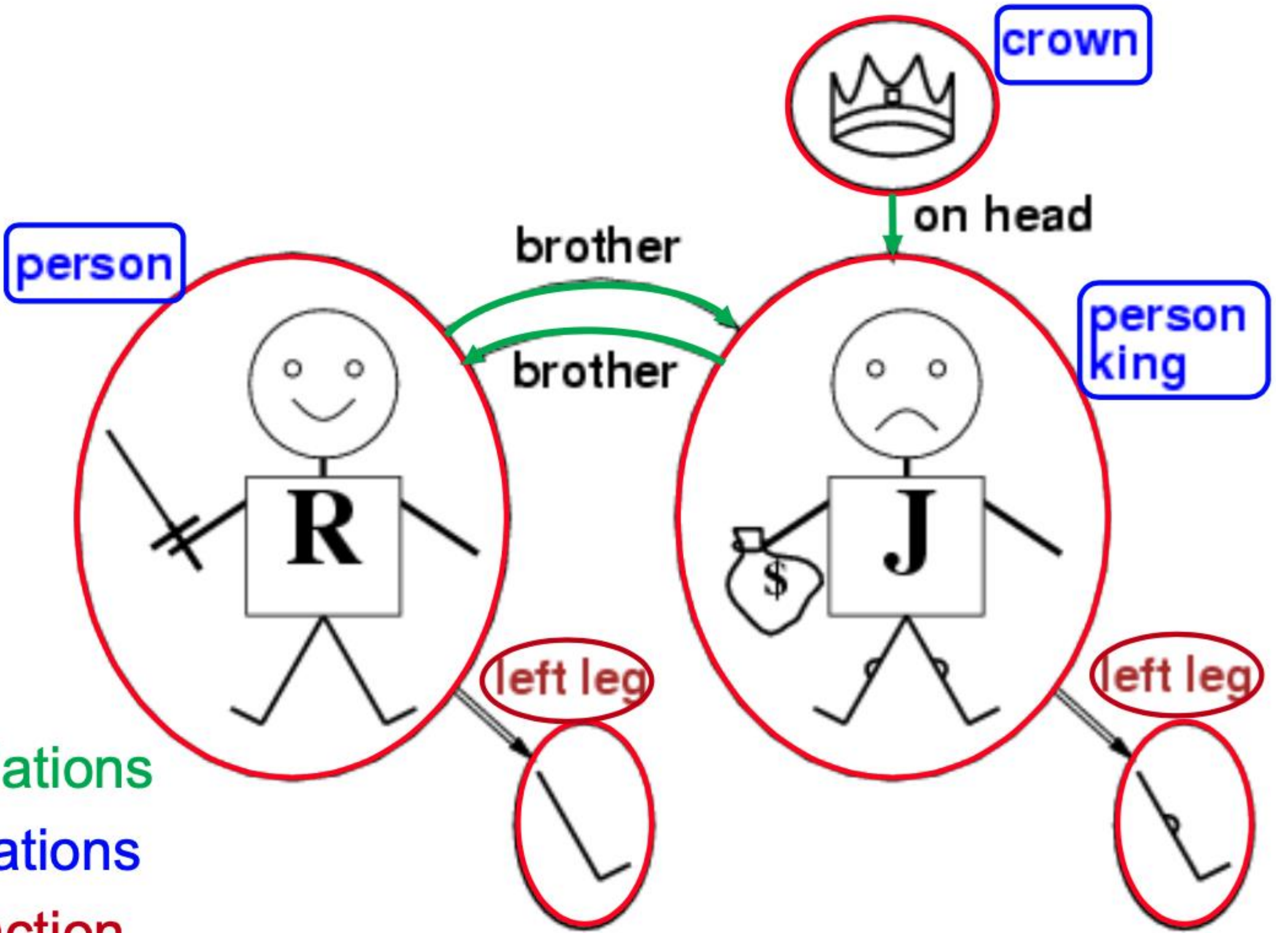
Types of Logic

Language	**Ontological Commitment** **(What exists in the world)**	**Epistemological Ontological 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 ∊ [0,1]
Fuzzy logic	Facts with degree of truth ∊ [0,1]	Known interval value

Models for a logic language

- Formal structures that constitute the possible worlds under consideration
- Each model links the vocabulary of sentences to elements of the possible world → determine the truth of any sentence
- Models for propositional logic link proposition symbols to predefined truth values.
- 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

Models for a logic language



- 5 objects
- 2 binary relations
- 3 unary relations
- 1 unary function

Symbols

- Constant symbols represent objects.
  - E.g., Richard, John, etc.
- Predicate symbols stand for relations.
  - E.g., Brother , OnHead, Person, King, and Crown, etc.
- Function symbols stand for functions.
  - E.g., LeftLeg
- Each predicate or function symbol comes with an arity that fixes the number of arguments.
  - E.g., Brother(x,y) → binary, LeftLeg(x) → unary, etc.
- These symbols begins with uppercase letters by convention.

Intended interpretation

- Interpretation specifies exactly which objects, relations and functions are referred to by the symbol.
- Each model includes an (intended) interpretation.

FOL syntax

$$
\begin{array}{rcl}
Sentence & \rightarrow & AtomicSentence \mid ComplexSentence \\
AtomicSentence & \rightarrow & Predicate \mid Predicate(Term, \ldots) \mid Term = Term \\
ComplexSentence & \rightarrow & (\ Sentence\ ) \mid [\ Sentence\ ] \\
 & \mid & \neg\ Sentence \\
 & \mid & Sentence \land Sentence \\
 & \mid & Sentence \lor Sentence \\
 & \mid & Sentence \ \Rightarrow\ Sentence \\
 & \mid & Sentence \ \Leftrightarrow\ Sentence \\
 & \mid & Quantifier\ Variable, \ldots\ Sentence \\
 & & \\
Term & \rightarrow & Function(Term, \ldots) \\
 & \mid & Constant \\
 & \mid & Variable
\end{array}
$$

FOL syntax

$$
\begin{array}{rcl}
\mathit{Quantifier} & \rightarrow & \forall \mid \exists \\
\mathit{Constant} & \rightarrow & A \mid X_1 \mid \mathit{John} \mid \cdots \\
\mathit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\mathit{Predicate} & \rightarrow & \mathit{True} \mid \mathit{False} \mid \mathit{After} \mid \mathit{Loves} \mid \mathit{Raining} \mid \cdots \\
\mathit{Function} & \rightarrow & \mathit{Mother} \mid \mathit{LeftLeg} \mid \cdots \\
\textsc{Operator Precedence} & : & \neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{array}
$$

Terms

- A term is a logical expression that refers to an object
  - Constant symbols: $John$
  - Function symbols: $LeftLeg(John)$
- A complex term is formed by a function symbol followed by a parenthesized list of terms as arguments.
- Term = function(term$_1$,...,term$_n$) or constant or variable

Atomic sentence

- An atomic sentences state facts by using a predicate symbol followed by a parenthesized list of terms.
- Atomic sentence = predicate(term$_1$,...,term$_n$)
- For example,
    - Brother(Richard, John)
    - Married(Father(Richard), Mother(John))
- It is true if the relation referred to by the predicate symbol holds among the objects referred to by the arguments

Complex sentences

- A complex sentences are made from atomic sentences using connectives.
- For example,
    - $\neg Brother(LeftLeg(Richard), John)$
    - $Brother(Richard, John) \land Brother(John, Richard)$
    - $King(Richard) \lor King(John)$
    - $\neg King(Richard) \Rightarrow King(John)$
- Syntax and semantics are the same as in propositional logic.

Quantifiers: Universal quantification

- ∀<variables> <sentence>
  - E.g., “Students of FIT are smart.”
  - $\forall x\ Student(x, FIT) \Rightarrow Smart(x)$
- ∀x P is true in a model m iff P is true with x being each possible object in the model.
- It is equivalent to the conjunction of instantiations of P.

Student(Lan, FIT) ⇒ Smart(Lan) ∧

Student(Tuan, FIT) ⇒ Smart(Tuan) ∧

Student(Long, FIT) ⇒ Smart(Long) ∧ …

Variables

- A variable is a term all by itself and able to serve as the argument of a function
    - E.g., in predicates $King(x)$ or in function $LeftLeg(x)$.
- Usually represented by lowercase letters
- A term with no variables is called a ground term.

A common mistake to avoid

- Typically, ⇒ is the main connective with ∀
  - The conclusion of the rule just for those objects for whom the premise is true
  - It says nothing at all about individuals for whom the premise is false.
- Too strong implication
  - $\forall x\ Student(x, FIT) \land Smart(x)$
  - It means “Everyone is a student of FIT and Everyone is smart.”

Quantifiers: Existential quantification

- ∃<variables> <sentence>
  - E.g., “Some students of FIT are smart.”
  - $\exists x\ Student(x, FIT) \land Smart(x)$
- ∃x P is true in a model m iff P is true with x being some possible object in the model.
- It is equivalent to the disjunction of instantiations of P.

Student(Lan, FIT) ∧ Smart(Lan) ∨

Student(Tuan, FIT) ∧ Smart(Tuan) ∨

Student(Long, FIT) ∧ Smart(Long) ∨ ...

Common mistake to avoid

- Typically, ∧ is the main connective with ∃
- Too weak implication
    - $\exists x\ Student(x, FIT) \Rightarrow Smart(x)$
    - It is true even with anyone who is not at FIT.

Nested quantifiers

- Multiple quantifiers enable more complex sentences.
- Simplest cases: Quantifiers are of the same type
  - $\forall x \forall y\ Brother(x, y) \Rightarrow Sibling(x, y)$
  - $\forall x \forall y\ Sibling(x, y) \Leftrightarrow Sibling(y, x)$
- Mixtures
  - $\forall x \exists y\ Loves(x, y)$ → “Everybody loves somebody.”
  - $\exists x \forall y\ Loves(y, x)$ → “There is someone loved by everyone.”
- The order of quantification is therefore very important.

Nested quantifiers

- Rule: The variable belongs to the innermost quantifier that mentions it.
    - $\forall x\ (Crown(x) \lor (\exists x\ Brother(Richard, x))$
- Workaround: Use different variable names with nested quantifier, e.g., $\exists z\ Brother(Richard, z)$

Quantifier duality

- De Morgan's rules

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

- For example

$\forall x \;\; Likes(x, IceCream)$ $\qquad$ $\neg\exists x \;\; \neg Likes(x, IceCream)$

$\exists x \;\; Likes(x, Brocoli)$ $\qquad$ $\neg\forall x \;\; \neg Likes(x, IceCream)$

Equality symbol =

- $term1 = term2$ is true under a given interpretation iff $term1$ and $term2$ refer to the same object
- The negation insists that two terms are not the same.

  $\exists x, y\ Brother(x, Richard) \land Brother(y, Richard) \land \neg(x = y)$

Practise

$\forall m, c\ Mother(c) = m \Leftrightarrow Female(m) \land Parent(m, c)$

$\forall w, h\ Husband(h, w) \Leftrightarrow Male(h) \land Spouse(h, w)$

$\forall x\ Male(x) \Leftrightarrow \neg Female(x)$

$\forall p,c\ Parent(p,c) \Leftrightarrow Child(c,p)$

$\forall g, c\ Grandparent(g, c) \Leftrightarrow \exists p\ Parent(g, p) \land Parent(p, c)$

$\forall x, y\ Sibling(x, y) \Leftrightarrow \neg(x = y) \land \exists p\ Parent(p, x) \land Parent(p, y)$

Practise

- Diagnostic rule → infer cause from effect
    - $\forall s\ Breezy(s) \Leftrightarrow \exists r\ Adjacent(r, s) \land Pit(r)$
- Causal rule → infer effect from cause
    - $\forall r\ Pit(r) \Leftrightarrow [\forall s\ Adjacent(r, s) \Rightarrow Breezy(s)]$

Universal Instantiation (UI)

- It is possible to infer any sentence obtained by substituting a ground term for the variable.
- Let $SUBST(\theta, \alpha)$ be the result of applying the substitution θ to the sentence α.
- Then the rule of Universal Instantiation is written

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

Universal Instantiation (UI)

$$\forall \boldsymbol{x}\ \boldsymbol{King}(\boldsymbol{x}) \land \boldsymbol{Greedy}(\boldsymbol{x}) \Rightarrow \boldsymbol{Evil}(\boldsymbol{x})$$

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

$King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)$

$King(Father(John)) \land Greedy(Father(John)) \Rightarrow Evil(Father(John))$

substitutions: $\{x/John\}$, $\{x/Richard\}$, $\{x/Father(John)\}$

Existential Instantiation (EI)

- It is possible to replace the variable by a single new constant symbol.
- The rule of Existential Instantiation is written

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

for any sentence α, variable v, and constant symbol k that does not appear elsewhere in KB.

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

C does not appear in $KB \rightarrow$ Skolem constant.

Universal / Existential Instantiation

- The UI rule can be applied many times to produce different consequences.
- The EI rule can be applied once, and then the existentially quantified sentence is discarded.
    - The new KB is not logically equivalent to the old but shown to be inferentially equivalent.

Generalized Modus Ponens (GMP)

- For atomic sentences p_i, p_i' and q, where there exists θ such that $SUBST(\theta, p_i') = SUBST(\theta, p_i)$, for all i

$$\frac{p_1', p_2', \dots, p_n', \quad (p_1, p_2, \dots, p_n \Rightarrow q)}{SUBST(\theta, q)}$$

- For example, p_1' is $King(John)$ $\quad$ p_1 is $King(x)$

  p_2' is $Greedy(y)$ $\quad$ p_2 is $Greedy(x)$

  θ is $\{x/John, y/John\}$ $\quad$ q is $Evil(x)$

  $SUBST(\theta, q)$ is $Evil(John)$

- All variables assumed universally quantified
- A lifted version of Modus Ponens → sound inference rule

Unification

- Find substitutions that make different logical expressions look identical

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

- For example,

p	q	$\boldsymbol{\theta}$
Knows(John, x)	Knows(John, Jane)	{x/Jane}
Knows(John, x)	Knows(y, Steve)	{x/Steve, y/John}
Knows(John, x)	Knows(y, Mother(y))	{x/Mother(John), y/John}
Knows(John, x)	Knows(x, Steve)	fail

- Standardizing apart eliminates overlap of variables:

  Knows(z, Steve)

Most General Unifier (MGU)

- $UNIFY(Knows(John, x), Knows(y, z)) = \theta$

  1. $\theta = \{y/John, x/z\}$

  2. $\theta = \{y/John, x/John, z/John\}$

- 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.

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

Unification algorithm

```
function UNIFY(x , y, θ) returns a substitution to make x and y identical
    inputs:     x , a variable, constant, list, or compound expression
                y, a variable, constant, list, or compound expression
                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
```

Unification algorithm

```
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

- A definite clause is a disjunctions of literals of which exactly one is positive.
- First-order literals can include variables, which are assumed to be universally quantified.
- Not every KB can be converted into a set of definite clauses due to the single-positive-literal restriction.

Forward Chaining

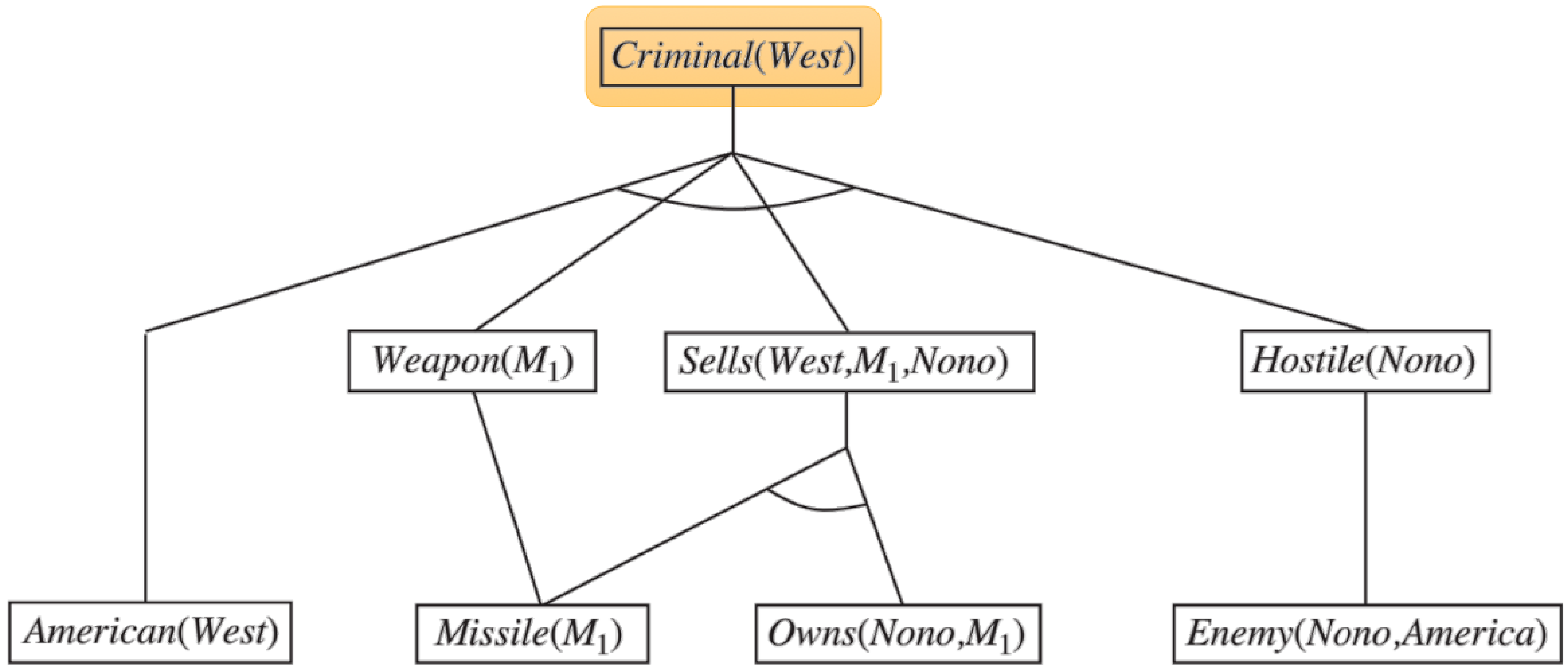
- “The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.”
- $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
- $\exists x\ Owns(Nono, x) \wedge Missile(x)$
  - $\boldsymbol{Owns(Nono, M_1)}$
  - $\boldsymbol{Missile(M_1)}$
- $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
- $Missile(x) \Rightarrow Weapon(x)$

  $\boldsymbol{American(West)}$
- $Enemy(x, America) \Rightarrow Hostile(x)$ $\boldsymbol{Enemy(Nono, America)}$

Forward Chaining

```
function FOL-FC-ASK($KB$,$\alpha$) returns a substitution or false
  inputs: $KB$, the knowledge base, a set of first-order definite clauses
          $\alpha$, the query, an atomic sentence
  local variables: new, the new sentences inferred on each iteration
  repeat until new is empty
    new ← { }
    for each rule in KB do
      $(p_1 \wedge \cdots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-VARIABLES(rule)
      for each $\theta$ such that SUBST($\theta$, $p_1 \wedge \cdots \wedge p_n$) = SUBST($\theta$,$p'_1 \wedge \cdots \wedge p'_n$ )
                                for some $p'_1, \ldots, p'_n$ in KB
        $q' \leftarrow$ SUBST($\theta$,$q$)
        if $q'$ does not unify with some sentence already in KB or new then
          add $q'$ to new
          $\varphi \leftarrow$ UNIFY($q'$,$\alpha$)
          if $\varphi$ is not fail then return $\varphi$
    add new to KB
  return false
```

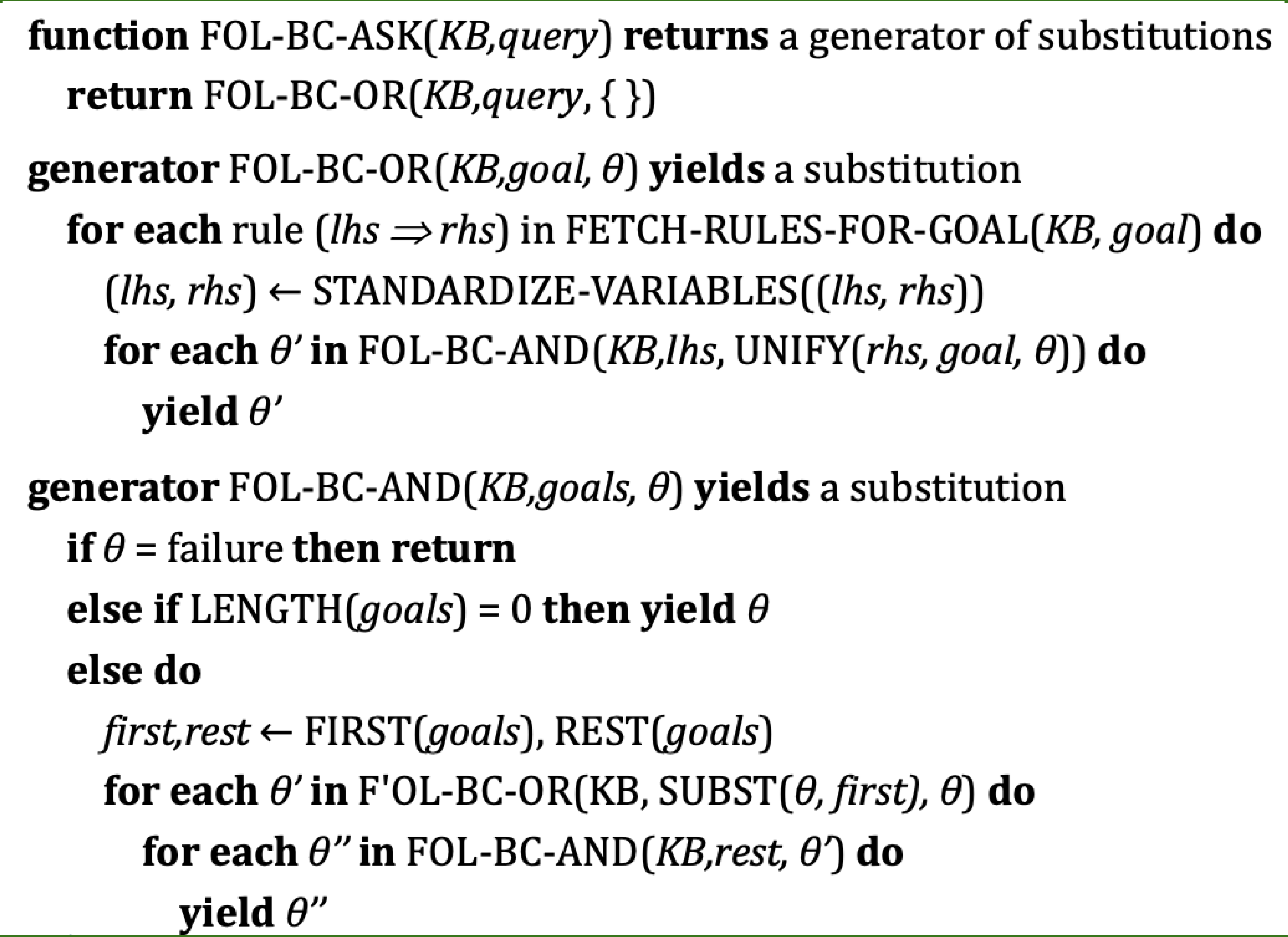
Forward Chaining



Forward Chaining

- Soundness?
    - YES, every inference is just an application of GMP.
- Completeness?
    - YES for definite clause knowledge bases.
- It answers every query whose answers are entailed by any KB of definite clauses.
- Terminate for Datalog in finite number of iterations
- Datalog = first-order definite clauses + no functions
- Entailment with definite clauses is semidecidable.

Backward chaining

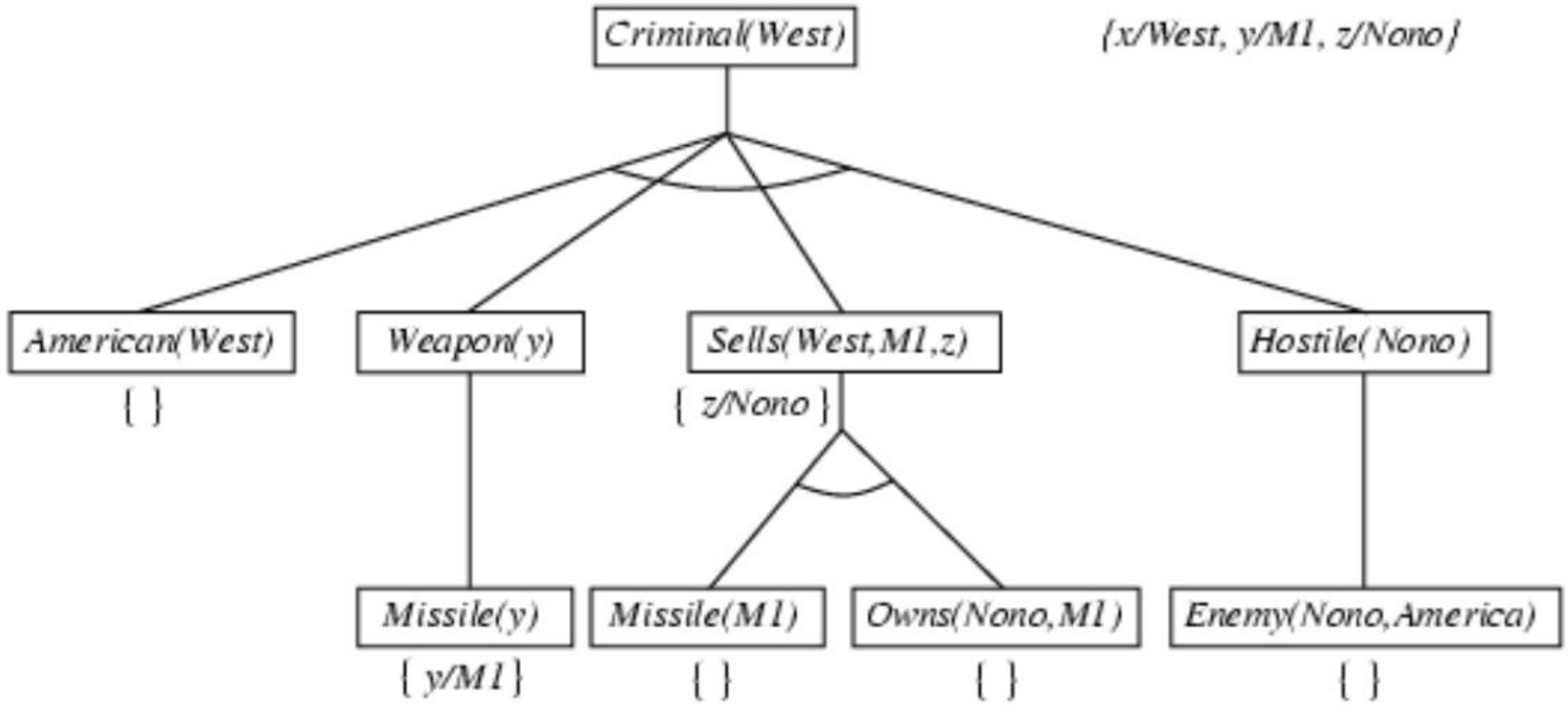


```
function FOL-BC-ASK(KB,query) returns a generator of substitutions
    return FOL-BC-OR(KB,query, { })

generator FOL-BC-OR(KB,goal, $\theta$) yields a substitution
    for each rule (lhs $\Rightarrow$ rhs) in FETCH-RULES-FOR-GOAL(KB, goal) do
        (lhs, rhs) $\leftarrow$ STANDARDIZE-VARIABLES((lhs, rhs))
        for each $\theta'$ in FOL-BC-AND(KB,lhs, UNIFY(rhs, goal, $\theta$)) do
            yield $\theta'$

generator FOL-BC-AND(KB,goals, $\theta$) yields a substitution
    if $\theta$ = failure then return
    else if LENGTH(goals) = 0 then yield $\theta$
    else do
        first,rest $\leftarrow$ FIRST(goals), REST(goals)
        for each $\theta'$ in F'OL-BC-OR(KB, SUBST($\theta$, first), $\theta$) do
            for each $\theta''$ in FOL-BC-AND(KB,rest, $\theta'$) do
                yield $\theta''$
```

Backward chaining



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 (success and failure)
  - Fix using caching of previous results (extra space)
- Widely used for logic programming

CNF for First-order logic

- First-order resolution requires that sentences be in CNF
- Every sentence of first-order logic can be converted into an inferentially equivalent CNF sentence.
- For example, the sentence

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

becomes, in CNF,

$$\neg American(x) \lor \neg Weapon(y) \lor \neg Sells(x, y, z) \lor \neg Hostile(z) \lor Criminal(x)$$

Conversion to CNF

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

- Eliminate implications

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

- Move ¬ in wards: $\neg\forall x\ p \equiv \exists x\ \neg p$, $\neg\exists x\ p \equiv \forall x\ \neg p$

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

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

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

- Standardize variables: each quantifier uses a different one

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

Conversion to CNF

- Skolemize: remove existential quantifiers by elimination
  - Simple case: translate $\exists x\ P(x)$ into $P(A)$, where A is a new constant.
    - However, $\forall x\ [Animal(A) \land \neg Loves(x,\ A)] \lor Loves(B,\ x)$ has an entirely different meaning.
  - The arguments of the Skolem function are all universally quantified variables in whose scope the existential quantifier appears.

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

- Drop universal quantifiers

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

Conversion to CNF

- Distribute ∨ over ∧

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

Resolution

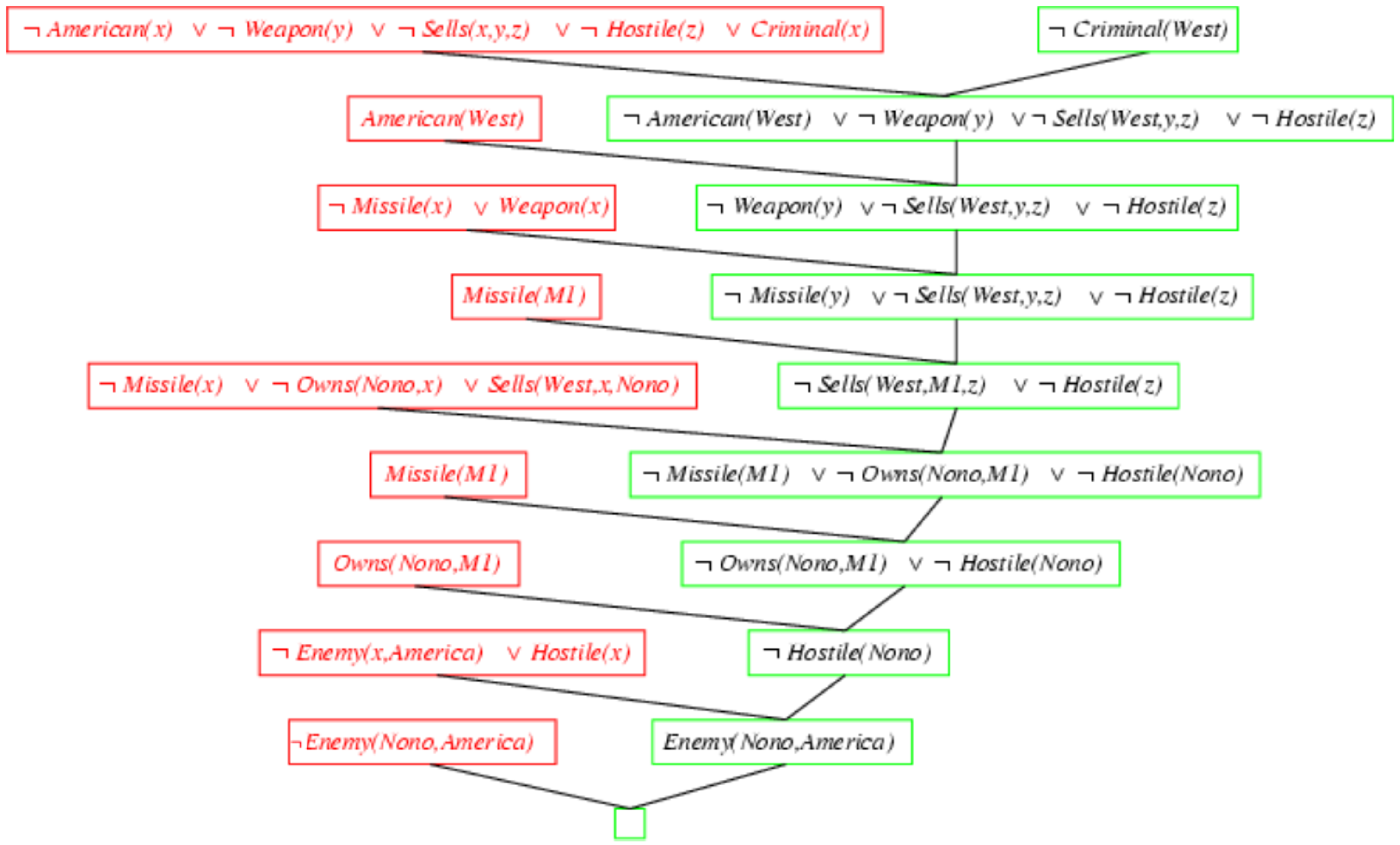
$$\frac{\begin{array}{c} l_1 \vee \cdots \vee l_k \\ m_1 \vee \cdots \vee m_n \end{array}}{\textbf{SUBST}(\boldsymbol{\theta}, l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

- where $UNIFY(l_i, \neg m_j) = \theta$

- For example,

$$\frac{\begin{array}{c} Animal(F(x)) \vee Loves(G(x), x) \\ \neg Loves(u, v) \vee \neg Kills(u, v) \end{array}}{Animal(F(x)) \vee \neg Kills(G(x), x)} \quad \theta = \{u/G(x), v/x\}$$

Resolution


¬ American(x) ∨ ¬ Weapon(y) ∨ ¬ Sells(x,y,z) ∨ ¬ Hostile(z) ∨ Criminal(x)
¬ Criminal(West)
American(West)
¬ American(West) ∨ ¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ Weapon(x)
¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ ¬ Owns(Nono,x) ∨ Sells(West,x,Nono)
¬ Sells(West,M1,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(M1) ∨ ¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
Owns(Nono,M1)
¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
¬ Enemy(x,America) ∨ Hostile(x)
¬ Hostile(Nono)
¬Enemy(Nono,America)
Enemy(Nono,America)

References

- Stuart Russell and Peter Norvig. 2009. Artificial Intelligence: A Modern Approach (3rd ed.). Prentice Hall Press, Upper Saddle River, NJ, USA.
- Lê Hoài Bắc, Tô Hoài Việt. 2014. Giáo trình Cơ sở Trí tuệ nhân tạo. Khoa Công nghệ Thông tin. Trường ĐH Khoa học Tự nhiên, ĐHQG-HCM.
- Nguyễn Ngọc Thảo, Nguyễn Hải Minh. 2020. Bài giảng Cơ sở Trí tuệ Nhân tạo. Khoa Công nghệ Thông tin. Trường ĐH Khoa học Tự nhiên, ĐHQG-HCM.