**22년 삼성 AI 전문가과정**
**6월 7일 화요일 5교시**
**장병탁**

# 인공지능

## 6차시 : Logical Inference

서울대학교 컴퓨터공학부
담당 교수: 장병탁

Seoul National University
Byoung-Tak Zhang

# Lecture Overview

# 인공지능

## 6차시 : Logical Inference


서울대학교 컴퓨터공학부
담당 교수: 장병탁

Seoul National University
Byoung-Tak Zhang


# Introduction

- **Knowledge-based agents** (Previous lecture)
  - Represent knowledge about the world
  - Deduce the actions to take
- **Representing knowledge using logic** (Previous lecture)
  - Propositional logic (PL)
  - First-order logic (FOL)
- **Inference in first-order logic** (This lecture)
  - Propositionalization
  - Forward chaining
  - Backward chaining
  - Resolution inference

# Knowledge in First-Order Logic: Colonel West Problem

## Colonel West 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.**

## First-order logic description of the problem

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

$Owns(Nono, M1)$

$Missile(M1)$

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

$Missile(x) \Rightarrow Weapon(x)$

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

$American(West)$

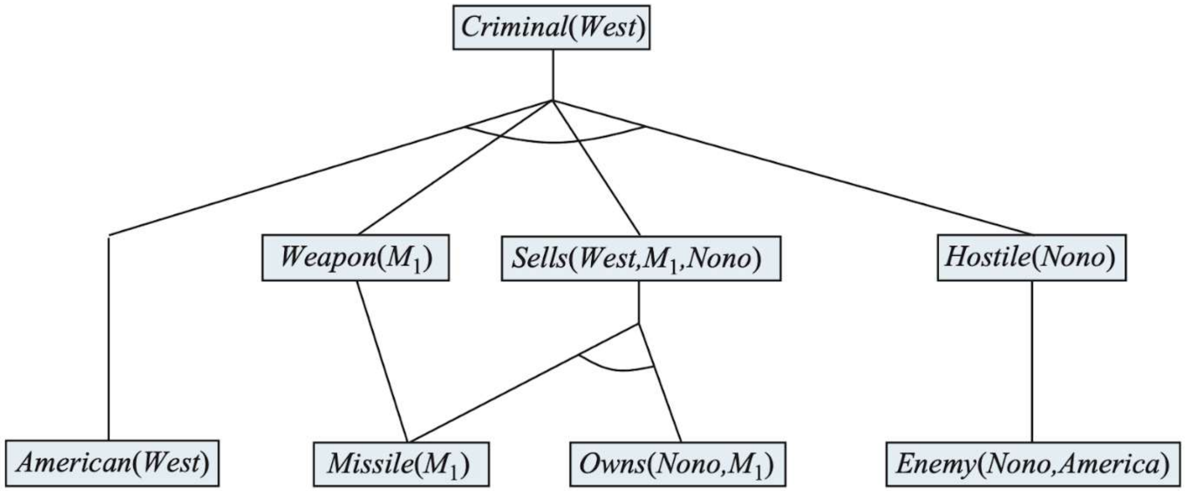
$Enemy(Nono, America)$

$\Rightarrow$ $Criminal(West)?$

# Inference in FOL: Proof Tree

- How can answer any answerable first-order logic question?
- There are four major ways to make inferences in FOL.
  - Propositionalization
  - Forward chaining
  - Backward chaining
  - Resolution inference

**Proving if Colonel West is a criminal**




출처: Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4rd Edition). Pearson

# Outline (Lecture 6)



Stuart Russell & Peter Norvig (2021), Artificial Intelligence: A Modern Approach (4th Edition)

# 6.1 Propositional vs. First-Order Inference

# 6.1 Propositional vs. First-Order Inference (1/5)

## Inference rules

- **Modus Ponens**
  - $\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$
    whenever any sentences of the form $\alpha \Rightarrow \beta$ and $\alpha$ are given, then the sentence $\beta$ can be inferred.
- **And-Elimination**
  - $\frac{\alpha \wedge \beta}{\alpha}$
- **Resolution**
  - $(\alpha_1, \ldots, \alpha_j, \ldots, \alpha_n), \ (\alpha_1, \ldots, \neg\alpha_j, \ldots, \alpha_n) \models (\alpha_1, \ldots, \alpha_{j-1}, \alpha_{j+1}, \ldots, \alpha_n)$

Both for PL & FOL

- **Universal instantiation (UI)**
- **Existential instantiation (EI)**

new for FOL

# 6.1 Propositional vs. First-Order Inference (2/5)

## Inference rules for quantifiers

$\forall x \ \text{Smart}(x)$

### 1) Universal instantiation (UI)

- We can infer any sentence $\alpha$ obtained by substituting a ground term $g$ for the variable $v$: $\dfrac{\forall v \ \alpha}{Subst(\{v/g\}, \alpha)}$
- $\forall x \ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

**yields**

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

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

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

# 6.1 Propositional vs. First-Order Inference (3/5)

## 2) Existential instantiation (EI)

- The variable $v$ in the sentence $\alpha$ is replaced by a single new constant symbol $k$:

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

$\boxed{\exists\, x\ \ \text{Smart}(x)}$

- From the sentence

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

we can infer the sentence

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

provided $C_1$ is a new constant symbol, called a Skolem constant.

- Inferentially equivalent (but not logically equivalent) in the sense that it is satisfiable exactly when the original KB is satisfiable.

# 6.1 Propositional vs. First-Order Inference (4/5)

## Reduction to Propositional Inference

- Suppose the KB contains just the following

  $\forall x\ King(x) \land Greedy(x) \Rightarrow Evil(x)$

  $King(John)$

  $Greedy(John)$

  $Brother(Richard, John)$

- Instantiating the universal sentence in all possible ways, $\{x/John\}$ and $\{x/Richard\}$, we have

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

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

- The new KB is propositionalized:

  $King(John), Greedy(John), Evil(John), King(Richard), etc.$

# 6.1 Propositional vs. First-Order Inference (5/5)

## Technique of Propositionalization

- First-order inference via propositionalization is complete– that is, any entailed sentence can be proved.
- What happens when the sentence is not entailed? We cannot tell.
- The question of entailment for first-order logic is semidecidable—that is, algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every non-entailed sentence.
- Cf. The halting problem for Turing machines.

# 6.2 Unification and Lifting

# 6.2 Unification and Lifting (1/4)

## Motivating Example

$$\forall x\ King(x) \wedge Greedy(x) \Longrightarrow Evil(x)$$

$$King(John)$$

$$\forall y\ Greedy(y)$$

- Propositionalization approach may generate

  $$King(Richard) \wedge Greedy(Richard) \Longrightarrow Evil(Richard)$$

  which does not match the KB and, thus, useless for proving $Evil(John)$.
- We can get the inference immediately if we can find a substitution $\theta$ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(John)$
- $\theta = \{x/John, y/John\}$ works

# 6.2 Unification and Lifting (2/4)

## Unification

- $Unify(p, q) = \theta$ where $p\theta = q\theta$ ($\theta$: most general unifier (MGU))
- Lifted inference rules require finding substitutions that make different logical expressions look identical. This process is called unification.

| $p$ | $q$ | $\theta$ |
|---|---|---|
| *Knows*(*John*, *x*) | *Knows*(*John*, *Jane*) | {*x* / *Jane* } |
| *Knows*(*John*, *x*) | *Knows*(*y*, *Bill*) | {*x* / *OJ*, *y* / *Jane* } |
| *Knows*(*John*, *x*) | *Knows*(*y*, *Mother*(*y*)) | {*y* / *Jone*, *x* / *Mother*(*John*)} |
| *Knows*(*John*, *x*) | *Knows*(*x*, *Elizabeth*) | *fail* |

# 6.2 Unification and Lifting (3/4)

## Generalized Modus Ponens (GMP)

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

*King*(*John*)
$\forall y$ *Greedy*(*y*)
$\forall x$ *King*(*x*) $\wedge$ *Greedy*(*x*) $\Longrightarrow$ *Evil*(*x*)

where $Subst(\theta, p_i') = Subst(\theta, p_i)$ or $p_i'\theta = p_i\theta$ for $\forall i$.

$p_1'$ is *King*(*John*)
$p_2'$ is *Greedy*(*y*)
$\theta$ is {*x* /*John*, *y*/*John*}
$Subst(\theta, q) = q\theta$ is *Evil*(*John*)

$p_1$ is *King*(*x*)
$p_2$ is *Greedy*(*x*)
$q$ is *Evil*(*x*)

GMP is a lifted version of MP. It raises MP from ground (variable-free) propositional logic to first-order logic.

# 6.2 Unification and Lifting (4/4)

## Soundness of GMP

$p \vDash p\theta$ by UI $\qquad p\theta = Subst(\theta, p)$

$p'_1, \ldots, p'_n \vDash p'_1\theta, \ldots, p'_n\theta$

$p_1 \wedge \ldots \wedge p_n \Rightarrow q \vDash p_1\theta \wedge \ldots \wedge p_n\theta \Rightarrow q\theta$

Now $\theta$ in GMP is defined as $p'_i\theta = p_i\theta$ for $\forall i$.

Thus, we have

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

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

# 6.3 Forward Chaining

# 6.3 Forward Chaining (1/6)

Inference methods for direct manipulation of FOL (unlike propositionalization)

**Example: Colonel West 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.

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

$Owns(Nono, M1)$

$Missile(M1)$

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

$Missile(x) \Rightarrow Weapon(x)$

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

$American(West)$

$Enemy(Nono, America)$

**First-order definite clauses**

**Datalog:** First-order definite clauses with no function symbols

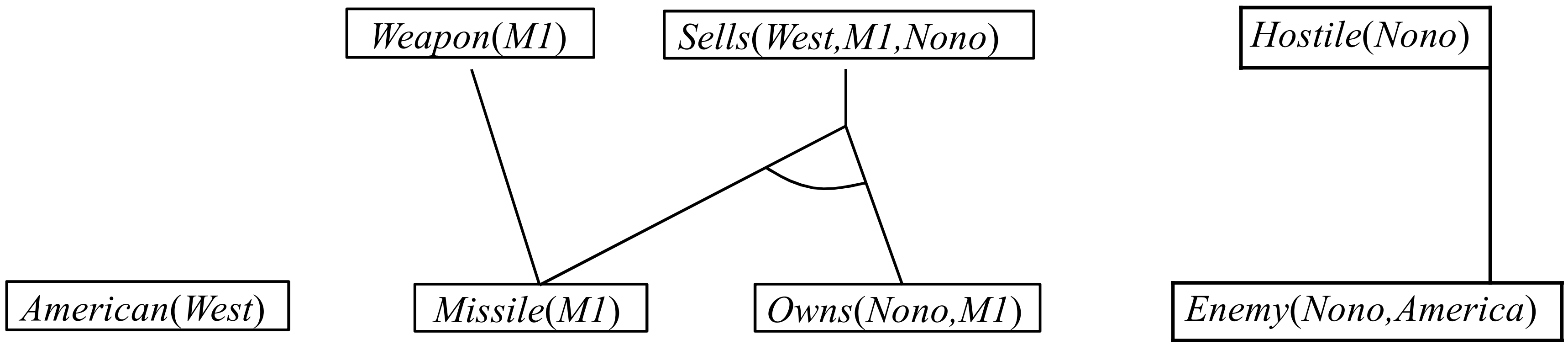
# 6.3 Forward Chaining (2/6)

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



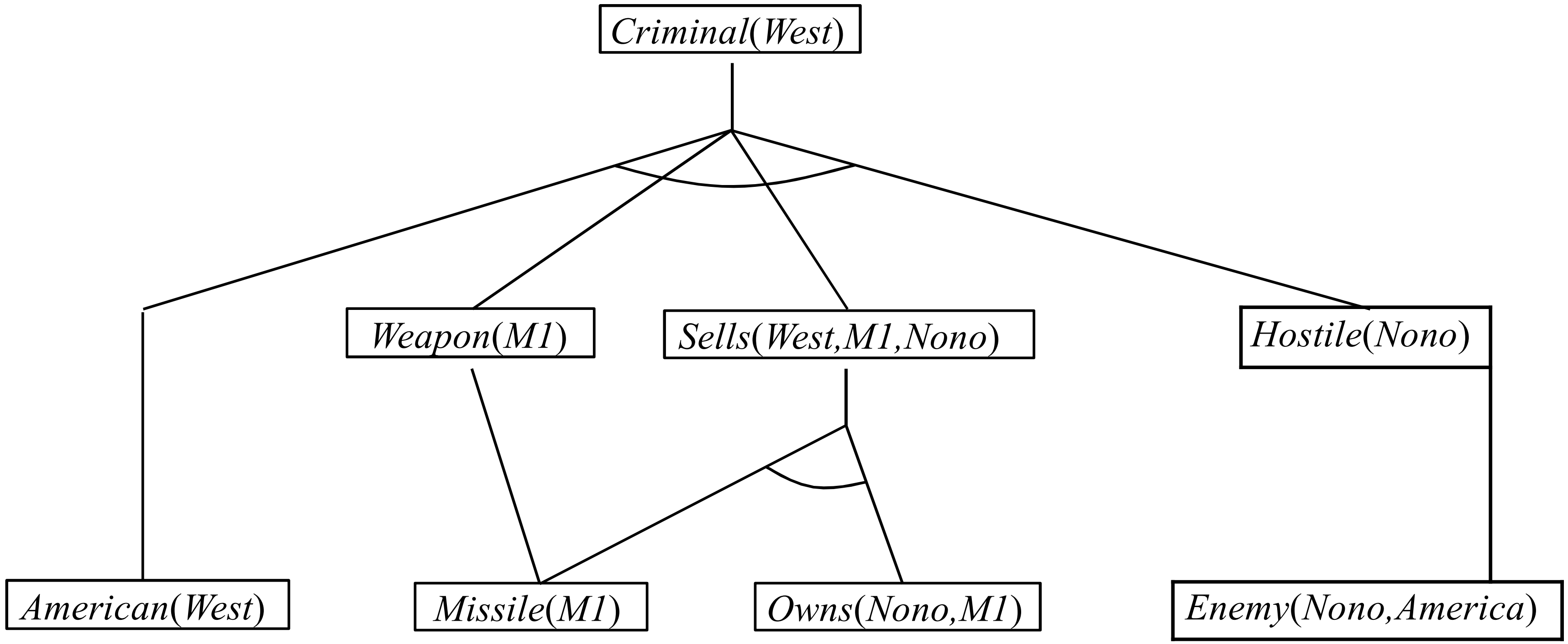
출처: Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4rd Edition). Pearson

# 6.3 Forward Chaining (3/6)




출처: Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4rd Edition). Pearson

# 6.3 Forward Chaining (4/6)




출처: Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4rd Edition). Pearson

# 6.3 Forward Chaining (5/6)

## Forward chaining algorithm

**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

  **while** *true* **do**
    $new \leftarrow \{\}$    *// The set of new sentences inferred on each iteration*
    **for each** *rule* **in** $KB$ **do**
      $(p_1 \land \ldots \land p_n \Rightarrow q) \leftarrow$ STANDARDIZE-VARIABLES(*rule*)
      **for each** $\theta$ such that SUBST($\theta, p_1 \land \ldots \land p_n$) = SUBST($\theta, p'_1 \land \ldots \land p'_n$)
          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*
          $\phi \leftarrow$ UNIFY($q', \alpha$)
          **if** $\phi$ is not *failure* **then return** $\phi$
    **if** $new = \{\}$ **then return** *false*
    add *new* to $KB$



출처: Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4rd Edition). Pearson

# 6.3 Forward Chaining (6/6)

## Properties of forward chaining

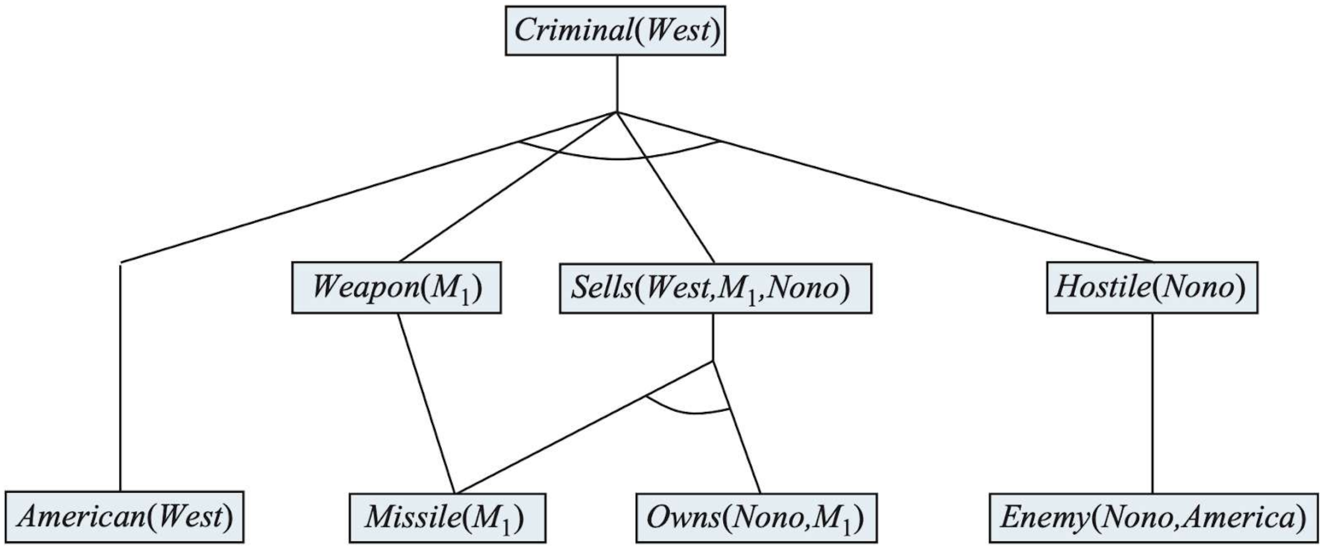
- Sound and complete for first-order definite clauses
  - Proof similar to propositional proof
- Datalog = first-order definite clauses + no functions (e.g., crime KB)
- FC terminates for Datalog in polynomial number of iterations
- May not terminate in general if $\alpha$ is not entailed
- This is unavoidable: entailment with definite clauses is semidecidable
- Forward chaining is widely used in deductive databases

# 6.4 Backward Chaining

# 6.4 Backward Chaining (1/4)

**The proof tree generated by forward chaining on the crime example.**

➢ The initial facts appear at the bottom level, facts inferred on the first iteration in the middle level, and facts inferred on the second iteration at the top level.






출처: Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4rd Edition). Pearson

# 6.4 Backward Chaining (2/4)

**A simple backward-chaining algorithm for first-order knowledge bases**

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

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

**function** FOL-BC-AND($KB$, $goals$, $\theta$) **returns** a substitution
  **if** $\theta = failure$ **then return**
  **else if** LENGTH($goals$) = 0 **then yield** $\theta$
  **else**
    $first, rest$ ← FIRST($goals$), REST($goals$)
    **for each** $\theta'$ **in** FOL-BC-OR($KB$, SUBST($\theta$, $first$), $\theta$) **do**
      **for each** $\theta''$ **in** FOL-BC-AND($KB$, $rest$, $\theta'$) **do**
        **yield** $\theta''$

출처: Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4rd Edition). Pearson

# 6.4 Backward Chaining (3/4)

## 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 (without improvements!) for logic programming

# 6.4 Backward Chaining (4/4)

## Logic programming (PROLOG)

- Algorithm = Logic + Control
- Prolog program = sets of definite clauses

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

```
append([], Y, Y).
append([A|X], Y, [A|Z]) :- append(X, Y, Z)
```

```
append(X, Y, [1,2])
→
X=[]       Y=[1,2];
X=[1]      Y=[2];
X=[1,2]    Y=[]
```

- Prolog uses database semantics, i.e. closed-world assumption and negation as failure
- Depth-first backward-chaining search

# 6.5 Resolution

# 6.5 Resolution (1/4)

**Resolution in full first-order**

$$A_1 \lor \ldots \lor A_k, m_1 \lor \ldots \lor m_n$$
$$(A_1 \lor \ldots \lor A_{i-1} \lor A_{i+1} \lor \ldots \lor A_k \lor m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n)\theta$$

where $Unify(Ai, \neg m_j) = \theta.$

For example,

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

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

# 6.5 Resolution (2/4)

**Conversion to CNF**

Everyone who loves all animals is loved by someone:

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

1. Eliminate biconditionals and implications

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

2. Move $\neg$ inwards: $\neg\forall x\ \ p\ \ \equiv\ \ \exists x\ \ \neg p,\quad \neg\exists x\ \ p\ \equiv\ \forall x\ \ \neg p$:

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

# 6.5 Resolution (3/4)

**Conversion to CNF (contd.)**

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:

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

5. Drop universal quantifiers:

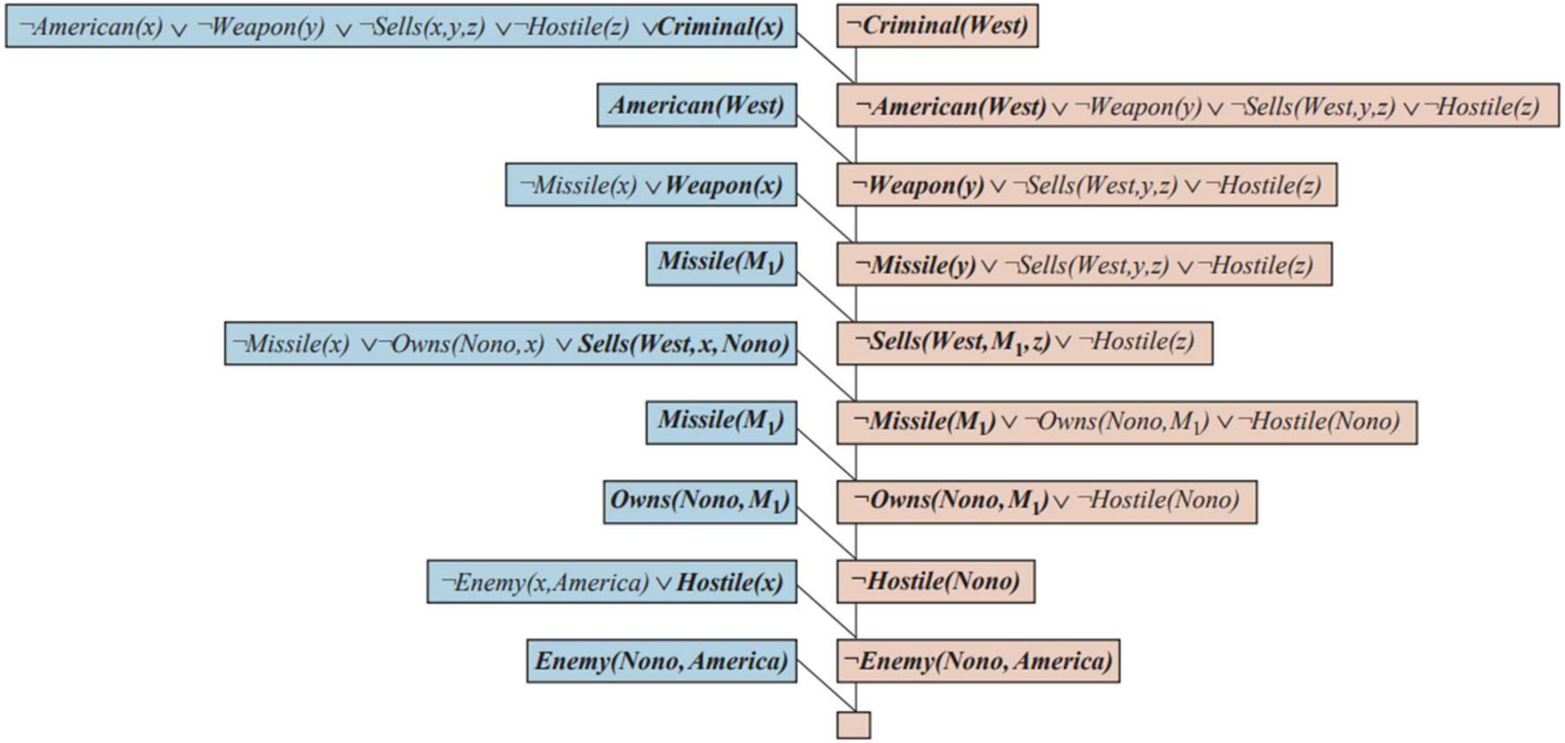
$$[Animal(F(x)) \land \neg Loves(x, F(x))\ ] \lor Loves(G(x), x)$$

6. Distribute over:

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

# 6.5 Resolution (4/4)

## Example proofs




출처: Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4rd Edition). Pearson

# Summary

- Instantiation is slow, unless the domain is small.
- The use of unification to identify appropriate substitutions for variables eliminates the instantiation step in FO proofs, making the process more efficient.
- A lifted version of Modus Ponens uses unification to provide a powerful inference rule, generalized Modus Ponens. The forward-chaining and backward-chaining algorithms apply this rule to sets of definite clauses.
- Forward chaining is used in deductive databases and production systems. Forward chaining is complete for Datalog.
- Prolog, unlike first-order logic, uses a closed world with the unique names assumption and negation as failure.
- The generalized resolution inference rule provides a complete proof system for first-order logic, using knowledge bases in CNF.