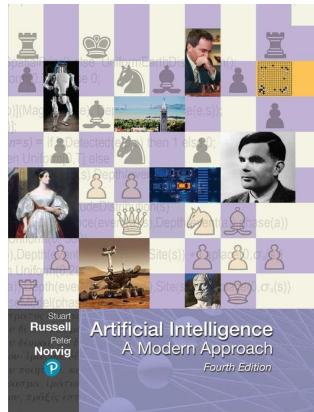


Artificial Intelligence: A Modern Approach

Fourth Edition



Copyright © 2021 Pearson Education, Inc. All Rights Reserved

Outline

- ◆ Reducing first-order inference to propositional inference
- ◆ Unification
- ◆ Generalized Modus Ponens
- ◆ Forward and backward chaining
- ◆ Logic programming
- ◆ Resolution

Chapter 9

Inference in first-order logic

1

P Pearson

© 2021 Pearson Education Ltd.

Chapter 9 2

Universal instantiation (UI)

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

$$\forall v \quad a$$

$$\underline{\text{Subst}(\{v/g\}, a)}$$

for any variable v and ground term g

E.g., $\forall x \quad \text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$ yields

$$\text{King}(J\,ohn) \wedge \text{Greedy}(J\,ohn) \Rightarrow \text{Evil}(J\,ohn)$$

$$\text{King}(R\ichard) \wedge \text{Greedy}(R\ichard) \Rightarrow$$

$$\text{Evil}(R\ichard)$$

$$\text{King}(F\ather(J\,ohn)) \wedge \text{Greedy}(F\ather(J\,ohn)) \Rightarrow \text{Evil}(F\ather(J\,ohn))$$

•

P Pearson

© 2021 Pearson Education Ltd.

Chapter 9 3

3

Existential instantiation (EI)

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

$$\exists v \quad a$$

$$\underline{\text{Subst}(\{v/k\}, a)}$$

E.g., $\exists x \quad \text{Crown}(x) \wedge \text{OnHead}(x, J\,ohn)$ yields

$$\text{Crown}(C_1) \wedge \text{OnHead}(C_1, J\,ohn)$$

provided C_1 is a new constant symbol, called a **Skolem**

constant. Another example: from $\exists x \quad d(x^y)/dy = x^y$ we obtain

$$d(e^y)/dy = e^y$$

provided e is a new constant symbol

P Pearson

© 2021 Pearson Education Ltd.

Chapter 9 4

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 is satisfiable iff the old KB was satisfiable

Existential instantiation contd.

Reduction to propositional inference

Suppose the KB contains just the following:

$$\begin{aligned} \forall x \text{King}(x) \wedge \text{Greedy}(x) &\Rightarrow \\ \text{Evil}(x) \quad \text{King}(J\,ohn) \\ \text{Greedy}(J\,ohn) \\ \text{Brother}(R\ichard, J\,ohn) \end{aligned}$$

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

$$\begin{aligned} \text{King}(J\,ohn) \wedge \text{Greedy}(J\,ohn) &\Rightarrow \text{Evil}(J\,ohn) \\ \text{King}(R\ichard) \wedge \text{Greedy}(R\ichard) &\Rightarrow \\ \text{Evil}(R\ichard) \quad \text{King}(J\,ohn) \\ \text{Greedy}(J\,ohn) \\ \text{Brother}(R\ichard, J\,ohn) \end{aligned}$$

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

P Pearson $\text{g}(J\,ohn), \text{Greedy}(J\,ohn), \text{Evil}(J\,ohn), \text{King}(R\ichard)$ etc.

Chapter 9 6

P Pearson

© 2021 Pearson Education Ltd.

Chapter 9 5

Reduction contd.

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., $\text{Father}(\text{Father}(\text{Father}(\text{John})))$

Theorem: Herbrand (1930). If a sentence a 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 a is entailed by this KB

Problem: works if a is entailed, loops if a is not entailed

P Pearson: Turing (1936), Church (1936), entailment in FOL is semidecidable

Problems with propositionalization

Propositionalization seems to generate lots of irrelevant sentences. E.g., from

$$\begin{aligned} \forall x \text{King}(x) \wedge \text{Greedy}(x) &\Rightarrow \\ \text{Evil}(x) \text{ King}(\text{John}) \\ \forall y \text{ Greedy}(y) \\ \text{Brother}(\text{Richard}, J \\ \text{ohn}) \end{aligned}$$

it seems obvious that $\text{Evil}(\text{John})$, but propositionalization produces lots of facts such as $\text{Greedy}(\text{Richard})$ that are irrelevant

With p k -ary predicates and n constants, there are $p \cdot n^k$

instantiations With function symbols, it gets much much worse!

Unification

We can get the inference immediately if we can find a substitution θ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(y)$

$\theta = \{x/\text{John}, y/\text{John}\}$ works

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

| p | q | θ |
|--------------------------------|--|----------|
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(\text{John},$ | |
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(\text{John}, \text{Jane})$ | |
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(y, M$ | |
| $\text{OJ})$ | $\text{Knows}(\text{John}, x)$ | |
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(y, M$ | |
| $\text{other}(y))$ | $\text{Knows}(\text{John}, x)$ | |
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(x, OJ)$ | |

P Pearson

© 2021 Pearson Education Ltd.

Chapter 9 8

Unification

We can get the inference immediately if we can find a substitution θ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(y)$

$\theta = \{x/\text{John}, y/\text{John}\}$ works

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

| p | q | θ |
|--------------------------------|--|---------------------|
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(\text{John},$ | $\{x/\text{Jane}\}$ |
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(\text{John}, \text{Jane})$ | |
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(y, M$ | |
| $\text{OJ})$ | $\text{Knows}(\text{John}, x)$ | |
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(y, M$ | |
| $\text{other}(y))$ | $\text{Knows}(\text{John}, x)$ | |
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(x, OJ)$ | |

P Pearson

© 2021 Pearson Education Ltd.

Chapter 9 10

Unification

We can get the inference immediately if we can find a substitution θ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(y)$

$\theta = \{x/\text{John}, y/\text{John}\}$ works

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

| p | q | θ |
|--------------------------------|--|---------------------|
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(\text{John},$ | $\{x/\text{Jane}\}$ |
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(\text{John}, \text{Jane})$ | |
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(y, M$ | |
| $\text{OJ})$ | $\text{Knows}(\text{John}, x)$ | |
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(y, M$ | |
| $\text{other}(y))$ | $\text{Knows}(\text{John}, x)$ | |
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(x, OJ)$ | |

P Pearson

© 2021 Pearson Education Ltd.

Chapter 9 10

Unification

We can get the inference immediately if we can find a substitution θ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(y)$

$\theta = \{x/\text{John}, y/\text{John}\}$ works

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

| p | q | θ |
|--------------------------------|--|---------------------|
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(\text{John},$ | $\{x/\text{Jane}\}$ |
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(\text{John}, \text{Jane})$ | |
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(y, M$ | |
| $\text{OJ})$ | $\text{Knows}(\text{John}, x)$ | |
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(y, M$ | |
| $\text{other}(y))$ | $\text{Knows}(\text{John}, x)$ | |
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(y, M$ | |
| $\text{other}(y))$ | $\text{Knows}(\text{John}, x)$ | |
| $\text{Knows}(\text{John}, x)$ | $\text{Knows}(x, OJ)$ | |

P Pearson

© 2021 Pearson Education Ltd.

Chapter 9 12

Unification

We can get the inference immediately if we can find a substitution θ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(J\text{ohn})$ and $\text{Greedy}(y)$

$\theta = \{x/\text{John}, y/J\text{ohn}\}$ works

Unify(a, β) = θ if $a\theta = \beta\theta$

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

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

Generalized Modus Ponens (GMP)

$$\frac{p_1^!, p_2^!, \dots, p_n^!, (p_1 \wedge \dots \wedge p_n \Rightarrow q)}{\theta} \quad \text{where } p_i^! \theta = p_i \theta \text{ for all } i$$

$p_1^!$ is $\text{King}(J\text{ohn})$
 $p_2^!$ is $\text{Greedy}(y)$
 θ is $\{x/\text{John}, y/J\text{ohn}\}$

GMP used with KB of **definite clauses** (exactly one positive literal)
All variables assumed universally quantified

Soundness of GMP

Need to show that

$$p_1^!, \dots, p_n^!, (p_1 \wedge \dots \wedge p_n \Rightarrow q) \models p\theta$$

provided that $p_i^! \theta = p_i \theta$ for all i

Lemma: For any definite clause p , we have $p \models p\theta$ by UI

$$1. (p_1 \wedge \dots \wedge p_n \Rightarrow q) \models (p_1 \wedge \dots \wedge p_n \Rightarrow q)\theta = (p_1 \theta \wedge \dots \wedge p_n \theta \Rightarrow q\theta)$$

$$2. p_1^!, \dots, p_n^! \models p_1^! \wedge \dots \wedge p_n^! \models p_1^! \theta \wedge \dots \wedge p_n^! \theta$$

3. From 1 and 2, $q\theta$ follows by ordinary Modus Ponens

Example knowledge base

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

Prove that Col. West is a criminal

Example knowledge base contd.

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

Example knowledge base contd.

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

Nono . . . has some missiles

Example knowledge base contd.

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

$$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$$

Nono . . . has some missiles, i.e., $\exists x \text{ Owns}(Nono, x) \wedge$

$$\text{Missile}(x): \text{Owns}(Nono, M) \text{ and } \text{Missile}(M)$$

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

Example knowledge base contd.

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

$$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$$

Nono . . . has some missiles, i.e., $\exists x \text{ Owns}(Nono, x) \wedge$

$$\text{Missile}(x): \text{Owns}(Nono, M) \text{ and } \text{Missile}(M)$$

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

$$\forall x \text{ Missile}(x) \wedge \text{Owns}(Nono, x) \Rightarrow \text{Sells}(West, x, Nono)$$

Missiles are weapons:

$$\text{Missile}(x) \Rightarrow \text{Weapon}(x)$$

An enemy of America counts as "hostile":

Example knowledge base contd.

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

$$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$$

Nono . . . has some missiles, i.e., $\exists x \text{ Owns}(Nono, x) \wedge$

$$\text{Missile}(x): \text{Owns}(Nono, M) \text{ and } \text{Missile}(M)$$

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

$$\forall x \text{ Missile}(x) \wedge \text{Owns}(Nono, x) \Rightarrow \text{Sells}(West, x, Nono)$$

Missiles are weapons:

Example knowledge base contd.

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

$$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$$

Nono . . . has some missiles, i.e., $\exists x \text{ Owns}(Nono, x) \wedge$

$$\text{Missile}(x): \text{Owns}(Nono, M) \text{ and } \text{Missile}(M)$$

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

$$\forall x \text{ Missile}(x) \wedge \text{Owns}(Nono, x) \Rightarrow \text{Sells}(West, x, Nono)$$

Missiles are weapons:

$$\text{Missile}(x) \Rightarrow \text{Weapon}(x)$$

An enemy of America counts as "hostile":

Example knowledge base contd.

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

$$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$$

Nono . . . has some missiles, i.e., $\exists x \text{ Owns}(Nono, x) \wedge$

$$\text{Missile}(x): \text{Owns}(Nono, M) \text{ and } \text{Missile}(M)$$

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

$$\forall x \text{ Missile}(x) \wedge \text{Owns}(Nono, x) \Rightarrow \text{Sells}(West, x, Nono)$$

Missiles are weapons:

$$\text{Missile}(x) \Rightarrow \text{Weapon}(x)$$

An enemy of America counts as "hostile":

$$\text{Enemy}(x, America) \Rightarrow \text{Hostile}(x)$$

West, who is American . . .

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

The country Nono, an enemy of America . . .

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

Forward chaining algorithm

```
function FOL-FC-Ask(KB, a) returns a substitution or false
repeat until new is empty
  new ← {}
  for each sentence r in KB do
    (p1 ∧ . . . ∧ pn ⇒ q) ← Standardize-Apart(r)
    for each θ such that (p1 ∧ . . . ∧ pn)θ = (pn ∧ . . .
      ∧ p')θ for some p1, . . . , p' in
      q' ← Subst(θ, q)
      if q' is not a renaming of a sentence already in KB or new then do
        add q' to new
        φ ← Unify(q', a)
        if φ is not fail then return φ
    add new to KB
  return false
```

Example knowledge base contd.

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

$$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$$

Nono . . . has some missiles, i.e., $\exists x \text{ Owns}(Nono, x) \wedge$

$$\text{Missile}(x): \text{Owns}(Nono, M) \text{ and } \text{Missile}(M)$$

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

$$\forall x \text{ Missile}(x) \wedge \text{Owns}(Nono, x) \Rightarrow \text{Sells}(West, x, Nono)$$

Missiles are weapons:

Example knowledge base contd.

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

$$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$$

Nono . . . has some missiles, i.e., $\exists x \text{ Owns}(Nono, x) \wedge$

$$\text{Missile}(x): \text{Owns}(Nono, M) \text{ and } \text{Missile}(M)$$

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

$$\forall x \text{ Missile}(x) \wedge \text{Owns}(Nono, x) \Rightarrow \text{Sells}(West, x, Nono)$$

Missiles are weapons:

$$\text{Missile}(x) \Rightarrow \text{Weapon}(x)$$

An enemy of America counts as "hostile":

$$\text{Enemy}(x, America) \Rightarrow \text{Hostile}(x)$$

West, who is American . . .

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

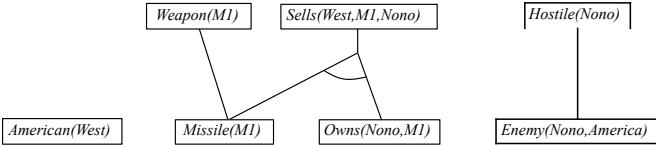
The country Nono, an enemy of America . . .

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

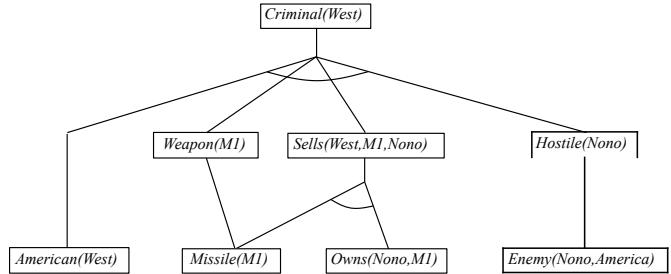
Forward chaining proof

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

Forward chaining proof



Forward chaining proof



Properties of forward chaining

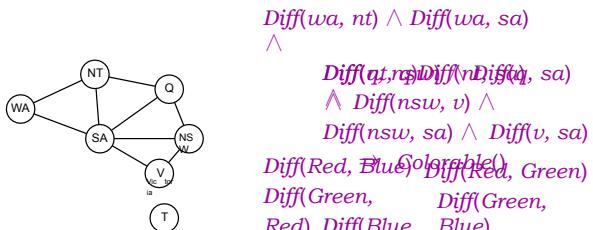
Sound and complete for first-order definite clauses (proof similar to propositional proof)

Datalog = first-order definite clauses + no functions (e.g., crime KB)
FC terminates for Datalog in poly iterations: at most $p \cdot n^k$ literals

May not terminate in general if a is not entailed

This is unavoidable: entailment with definite clauses is semidecidable

Hard matching example



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

Efficiency of forward chaining

Simple observation: no need to match a rule on iteration k
if a premise wasn't added on iteration $k-1$

⇒ 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 $\text{Missile}(x)$ retrieves $\text{Missile}(M_1)$

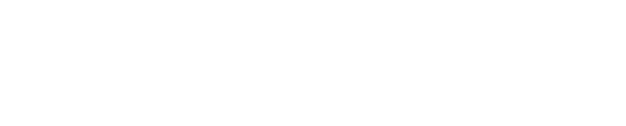
Matching conjunctive premises against known facts is NP-hard Forward chaining is widely used in deductive databases

Backward chaining algorithm

```
function FOL-BC-Ask(KB, goals, θ) returns a set of substitutions
  inputs: KB, a knowledge base
  inputs: goals, a list of conjuncts forming a query (θ already applied)
  inputs: θ, the current substitution, initially the empty substitution {}
  local variables: answers, a set of substitutions, initially empty
  if goals is empty then return {θ}
  q1 ← Subst(θ, First(goals))
  for each sentence r in KB
    where Standardize-Apart(r) = (p1 ∧ ... ∧ pn ⇒ q)
    and θ1 ← Unify(q, q1) succeeds
    new_goals ← [p1, ..., pn]Rest(goals)
    answers ← FOL-BC-Ask(KB, new_goals, Compose(θ1, θ)) ∪ answers
  return answers
```

Backward chaining example

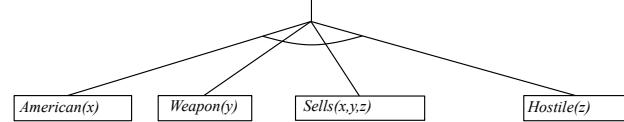
$Criminal(West)$



Backward chaining example

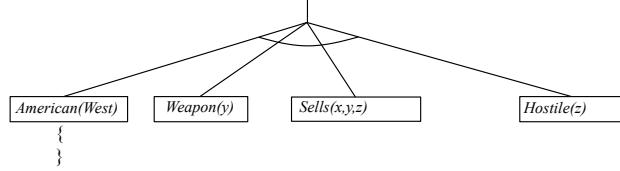
$Criminal(West)$

$\{x/West\}$



Backward chaining example

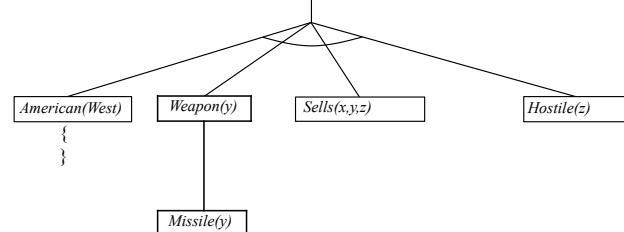
$Criminal(West)$



Backward chaining example

$Criminal(West)$

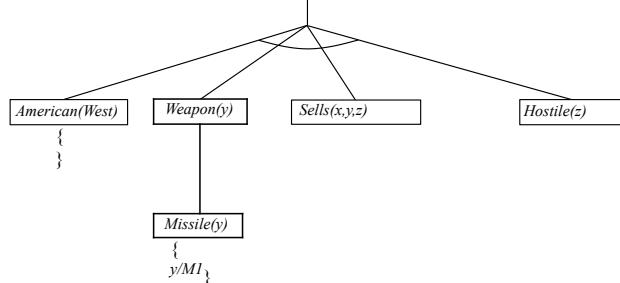
$\{x/West\}$



Backward chaining example

$Criminal(West)$

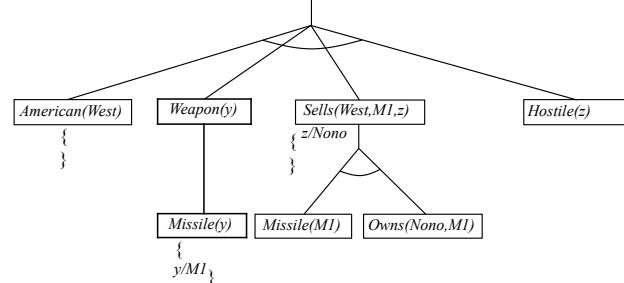
$\{x/West, y/MI\}$



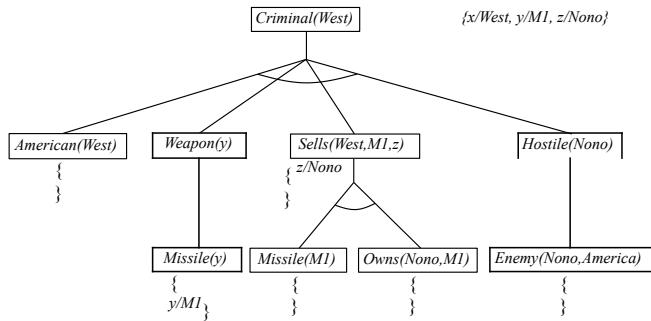
Backward chaining example

$Criminal(West)$

$\{x/West, y/MI, z/Nono\}$



Backward chaining example



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](#)

Logic programming

Sound bite: computation as inference on logical

KBs [Logic programming](#)

1. Identify problem
2. Assemble information
3. Tea break
4. Encode information in KB
5. Encode problem instance as facts
6. Ask queries
7. Find false facts

Should be easier to debug [Capital\(Needy one, US\)](#) than $x = x + 2$!

Ordinary programming

- Identify problem
- Assemble information
- Figure out solution
- Program solution
- Encode problem instance as data
- Apply program to data
- Debug procedural errors

Prolog systems

Basis: backward chaining with Horn clauses + bells & whistles

Widely used in Europe, Japan (basis of 5th Generation project) Compilation techniques ⇒ approaching a billion LIPS

Program = set of clauses = head :- literal₁, ..., literal_n.

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

Efficient unification by [open coding](#)

Efficient retrieval of matching clauses by direct

linking Depth-first, left-to-right backward chaining

Built-in predicates for arithmetic etc., e.g., X is Y*Z+3

Closed-world assumption ("negation as

failure") e.g., given alive(X) :- not

dead(X). alive(joe) succeeds if

dead(joe) fails

Prolog examples

Depth-first search from a start state X:

```
dfs(X) :- goal(X).
dfs(X) :- successor(X,S),dfs(S).
```

No need to loop over S: successor succeeds for each

Appending two lists to produce a third:

```
append([], Y, Y).
```

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

query: append(A,B,[1,2]) ?

answers: A=[] B=[1,2]
A=[1] B=[2]
A=[1,2] B=[]

Resolution: brief summary

Full first-order version:

$$\frac{(_1 \vee \dots \vee _i \vee \dots \vee _k \vee m_1 \vee \dots \vee m_n)}{m_n \theta}$$

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

For example,

$\neg \text{Rich}(x) \vee$

$\text{Unhappy}(x)$

$\text{Rich}(\text{Ken})$

$\text{Unhappy}(\text{Ken})$

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

Apply resolution steps to [CNF \(KB \$\wedge \neg a\$ \)](#); complete for FOL

Conversion to CNF

Everyone who loves all animals is loved by someone:

$$\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Loves}(x, y)] \equiv \exists x \text{ Loves}(y, x)$$

1. Eliminate biconditionals and implications

1. Eliminate biconditionals and implications
2. Move \neg inwards: $\neg \forall x, p \equiv \exists x \neg p$, $\neg \exists x, p \equiv \forall x \neg p$
3. Standardize variables: each quantifier should use a different one

Conversion to CNF contd.

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

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

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

$$\forall x [\text{Animal}(F(x)) \wedge \neg \text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x)$$

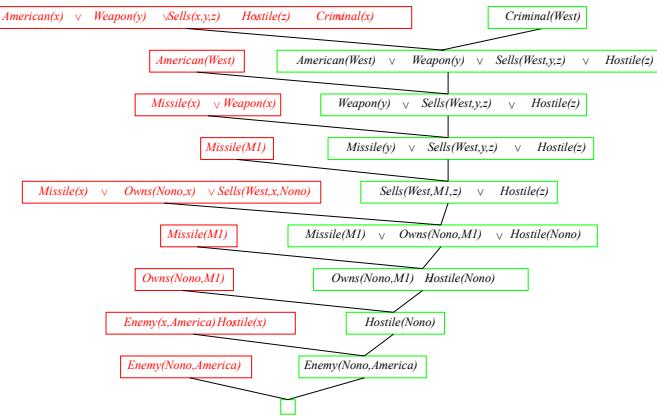
5. Drop universal quantifiers:

$$[\text{Animal}(F(x)) \wedge \neg \text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x)$$

6. Distribute \wedge over \vee :

$$[\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)] \wedge [\neg \text{Loves}(x, F(x)) \vee \text{Loves}(G(x), x)]$$

Resolution proof: definite clauses



Gödel's Incompleteness Theorem

- There are true arithmetic sentences that cannot be proved
- For any set of true sentences of number theory, and in particular any set of basic axioms, there are other true sentences that cannot be proved from those axioms.
- We can never prove all the theorems of mathematics within any given system of axioms.

Resolution strategies

- **Unit preference:** prefers to do resolutions where one of the sentences is a single literal (unit clause)
- **Set of support:** every resolution step involves at least one element of a special set of clauses
- **Input resolution:** every resolution combines one of the KB input sentences with other sentences
- **Subsumption:** eliminates all sentences that are subsumed by KB sentences
- **Learning:** learning from experience (machine learning)

Summary

- **Unification** identifies appropriate substitutions for variables eliminates the instantiation step in first-order proofs, making the process more efficient in many cases
- **Forward chaining** is used in deductive databases, where it can be combined with relational database operations. It is also used in production systems
- **Backward chaining** is used in logic programming systems, which employ sophisticated compiler technology to provide very fast inference
- **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 conjunctive normal form.