Outline

- Unification
- Generalized Modus Ponens
- Forward and backward chaining
- Logic programming
- Resolution

Unification

Unification

- We can get the inference immediately if we can find a substitution θ such that King(x) and Greedy(x) match King(John) and Greedy(y)
- $\theta = \{x/John, y/John\}$ works
- $\text{UNIFY}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

p	q	θ
Knows(John, x)	Knows(John, Jane)	
Knows(John, x)	Knows(y, OJ)	
Knows(John, x)	Knows(y, Mother(y))	
Knows(John, x)	Knows(x, OJ)	

Unification

- We can get the inference immediately if we can find a substitution θ such that King(x) and Greedy(x) match King(John) and Greedy(y)
- $\theta = \{x/John, y/John\}$ works
- $\textsc{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

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

Unification

- We can get the inference immediately if we can find a substitution θ such that King(x) and Greedy(x) match King(John) and Greedy(y)
- $\theta = \{x/John, y/John\}$ works
- $\textsc{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

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

Unification

- We can get the inference immediately if we can find a substitution θ such that King(x) and Greedy(x) match King(John) and Greedy(y)
- $\theta = \{x/John, y/John\}$ works
- $\text{UNIFY}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

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

Unification

- We can get the inference immediately if we can find a substitution θ such that King(x) and Greedy(x) match King(John) and Greedy(y)
- $\theta = \{x/John, y/John\}$ works
- $\textsc{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

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

Unification

- We can get the inference immediately if we can find a substitution θ such that King(x) and Greedy(x) match King(John) and Greedy(y)
- $\theta = \{x/John, y/John\}$ works
- $\text{UNIFY}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

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

- How to compute “unifiers” for two given terms/literals?

Unification Algorithm

Given two terms s and t.

- A Unifier for s and t is a **substitution** σ, such that $s\sigma = t\sigma$
- A unifier σ is most general, if for any other unifier τ of s and t, there exists a substitution ρ, such that $\rho(\sigma(s)) = \tau(s)$ and $\rho(\sigma(t)) = \tau(t)$.
- We give a unification algorithm as a set $\mathcal{R}$ of rules $\frac{U, E}{U', E'}$, which transforms sets of equations U and E into sets of rules U' and E'.
- The set U is always in solved form, i.e. all equations in U are of the form $x = t$, where x is a variable and x does not occur in t (gives directly the unifier).

Unification Algorithm

A simple unification algorithm:

- **Deleting** $\dfrac{U, E \cup \{t = t\}}{U, E}$
- **Decomposition** $\dfrac{U, E \cup f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)}{U, E \cup \{s_1 = t_1, \ldots, s_n = t_n\}}$
- **Orientation** $\dfrac{U, E \cup \{t = x\}}{U, E \cup \{x = t\}}$

  if t is not a variable
- **Substitution** $\dfrac{U, E \cup \{x = t\}}{U\{t/x\} \cup \{x = t\}, E\{t/x\}}$

  if x does not occur in t

Unification Algorithm contd

- **Occur-Check** $\dfrac{U, E \cup \{x = t\}}{\bot, \bot}$

  if x occurs in t

- **Clash** $\dfrac{U, E \cup f(s_1, \ldots, s_n) = g(t_1, \ldots, t_n)}{\bot, \bot}$ if $f \neq g$.

Two terms s and t are unifiable if using the rules $\mathcal{R}$ we can derive from $\{\}, \{s = t\}$ a pair $U, \{\}$.

The unifier contained in U is a most general unifier

Examples: $f(x, g(a, y)) \stackrel{?}{=} f(h(y), g(a, a))$

$f(x, g(a, y)) \stackrel{?}{=} f(h(y), g(a, h(x)))$

$f(x, g(a, x)) \stackrel{?}{=} f(h(y), g(a, a))$.

Generalizing Modus Ponens

Generalized Modus Ponens (GMP)

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

where $p_i'\theta = p_i\theta$ for all i

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

- GMP used with KB of definite clauses (**exactly** one positive literal)
- All variables implicitely universally quantified on the level of each definite clause
  Let $x_1, \ldots, x_n$ be all free variables in $(p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q$;
  So implicitly we have: $\forall x_1, \ldots, x_n \ \ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q$
- In general, need to rename shared variables before.

Soundness of GMP

Need to show that

$$p_1', \ldots, p_n', \quad (p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models q\theta$$

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

$\Longrightarrow$ Prove that Col. West is a criminal

Example knowledge base contd.

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

Example knowledge base contd.

- . . . it is a crime for an American to sell weapons to hostile nations:
  $\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$
- Nono . . . has some missiles, i.e., $\exists x\, \text{Owns}(\text{Nono}, x) \land \text{Missile}(x)$:
  $\text{Owns}(\text{Nono}, M_1)$ and $\text{Missile}(M_1)$

Example knowledge base contd.

- . . . it is a crime for an American to sell weapons to hostile nations:
  $\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$
- Nono . . . has some missiles, i.e., $\exists x\ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x)$:
  $\text{Owns}(\text{Nono}, M_1)$ and $\text{Missile}(M_1)$
- . . . all of its missiles were sold to it by Colonel West
  $\forall x\ \ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$

Example knowledge base contd.

- . . . it is a crime for an American to sell weapons to hostile nations:
  $\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$
- Nono . . . has some missiles, i.e., $\exists x\ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x)$:
  $\text{Owns}(\text{Nono}, M_1)$ and $\text{Missile}(M_1)$
- . . . all of its missiles were sold to it by Colonel West
  $\forall x\ \ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$
- Missiles are weapons:
  $\text{Missile}(x) \Rightarrow \text{Weapon}(x)$

Example knowledge base contd.

- . . . it is a crime for an American to sell weapons to hostile nations:
  $\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$
- Nono . . . has some missiles, i.e., $\exists x\ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x)$:
  $\text{Owns}(\text{Nono}, M_1)$ and $\text{Missile}(M_1)$
- . . . all of its missiles were sold to it by Colonel West
  $\forall x\ \ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$
- Missiles are weapons:
  $\text{Missile}(x) \Rightarrow \text{Weapon}(x)$
- An enemy of America counts as “hostile”:
  $\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$

Example knowledge base contd.

- . . . it is a crime for an American to sell weapons to hostile nations:
  $\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$
- Nono . . . has some missiles, i.e., $\exists x\ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x)$:
  $\text{Owns}(\text{Nono}, M_1)$ and $\text{Missile}(M_1)$
- . . . all of its missiles were sold to it by Colonel West
  $\forall x\ \ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$
- Missiles are weapons:
  $\text{Missile}(x) \Rightarrow \text{Weapon}(x)$
- An enemy of America counts as “hostile”:
  $\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$
- West, who is American . . . $\text{American}(\text{West})$

Example knowledge base contd.

- . . . it is a crime for an American to sell weapons to hostile nations:
  $\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$
- Nono . . . has some missiles, i.e., $\exists x\ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x)$:
  $\text{Owns}(\text{Nono}, M_1)$ and $\text{Missile}(M_1)$
- . . . all of its missiles were sold to it by Colonel West
  $\forall x\ \ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$
- Missiles are weapons:
  $\text{Missile}(x) \Rightarrow \text{Weapon}(x)$
- An enemy of America counts as “hostile”:
  $\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$
- West, who is American . . . $\text{American}(\text{West})$
- The country Nono, an enemy of America . . .
  $\text{Enemy}(\text{Nono}, \text{America})$

Forward chaining algorithm

```
function FOL-FC-Ask(KB, α) 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 = (p'1 ∧ ... ∧ p'n)θ
                                for some p'1, ..., p'n in KB
                q' ← Subst(θ, q)
                if q' not a renaming of a sentence already in KB or new then do
                    add q' to new
                    φ ← Unify(q', α)
                    if φ is not fail then return φ
        add new to KB
    return false
```

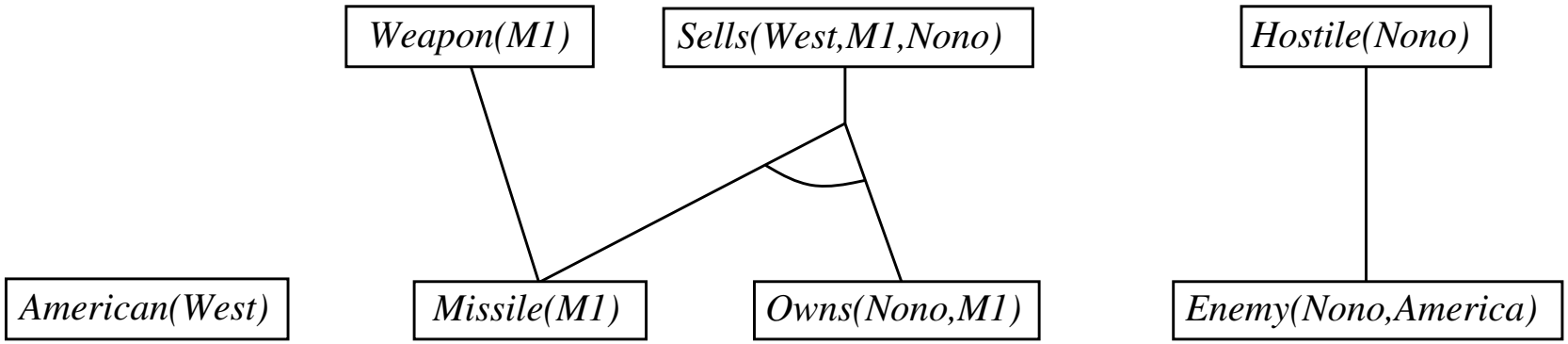
- Standardize-Apart(r): Rename free variables of r to new variables

Forward chaining proof

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

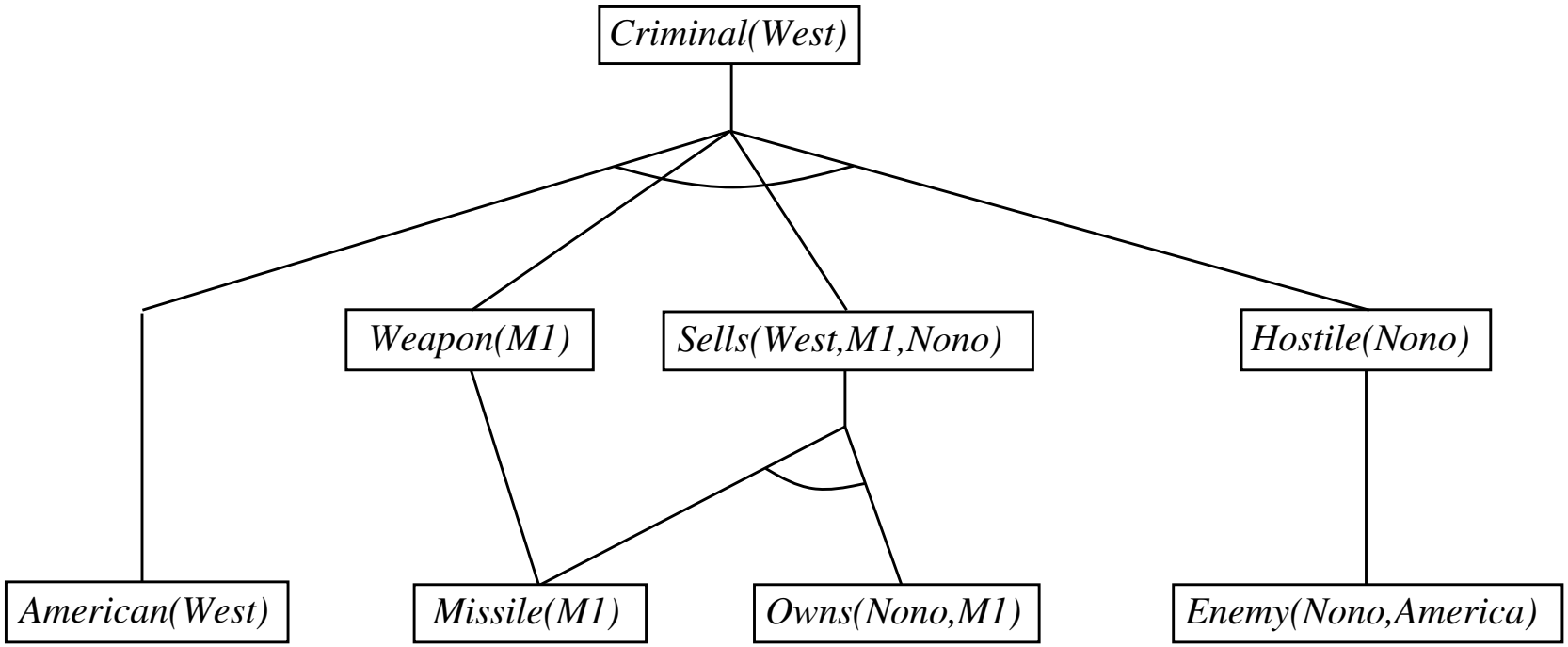
- $\text{Missile}(x) \Rightarrow \text{Weapon}(x)$
- $\text{Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$
- $\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$
- $\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$

Forward chaining proof



- $\text{Missile}(x) \Rightarrow \text{Weapon}(x)$
- $\text{Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$
- $\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$
- $\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$

Forward chaining proof



- $\text{Missile}(x) \Rightarrow \text{Weapon}(x)$
- $\text{Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$
- $\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$
- $\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$

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 α is not entailed
- This is unavoidable: entailment with definite clauses is semidecidable
- Example: Infinitely many facts are generated from
  $\mathsf{Nat}(0), \forall x \;\; \mathsf{Nat}(x) \Rightarrow \mathsf{Nat}(s(x))$.

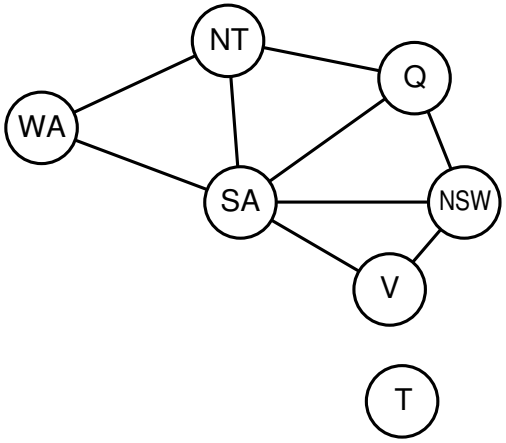
Efficiency of forward chaining

- 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 $\mathrm{Missile}(x)$ retrieves $\mathrm{Missile}(M_1)$
- Matching conjunctive premises against known facts is NP-hard
- Forward chaining is widely used in deductive databases

Hard matching example



$$Diff(\mathsf{wa},\mathsf{nt}) \wedge Diff(\mathsf{wa},\mathsf{sa}) \wedge$$
$$Diff(\mathsf{nt},\mathsf{q})\,Diff(\mathsf{nt},\mathsf{sa}) \wedge$$
$$Diff(\mathsf{q},\mathsf{nsw}) \wedge Diff(\mathsf{q},\mathsf{sa}) \wedge$$
$$Diff(\mathsf{nsw},\mathsf{v}) \wedge Diff(\mathsf{nsw},\mathsf{sa}) \wedge$$
$$Diff(\mathsf{v},\mathsf{sa}) \Rightarrow \mathsf{Colorable}()$$

$$Diff(\mathsf{Red},\mathsf{Blue}) \quad Diff(\mathsf{Red},\mathsf{Green})$$
$$Diff(\mathsf{Green},\mathsf{Red}) \quad Diff(\mathsf{Green},\mathsf{Blue})$$
$$Diff(\mathsf{Blue},\mathsf{Red}) \quad Diff(\mathsf{Blue},\mathsf{Green})$$

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

(WA = West Australia, SA = South Australia, NT = Northern Territory, Q = Queensland, NSW = New South Wales, V = Victoria, T = Tasmania)

Backward chaining algorithm

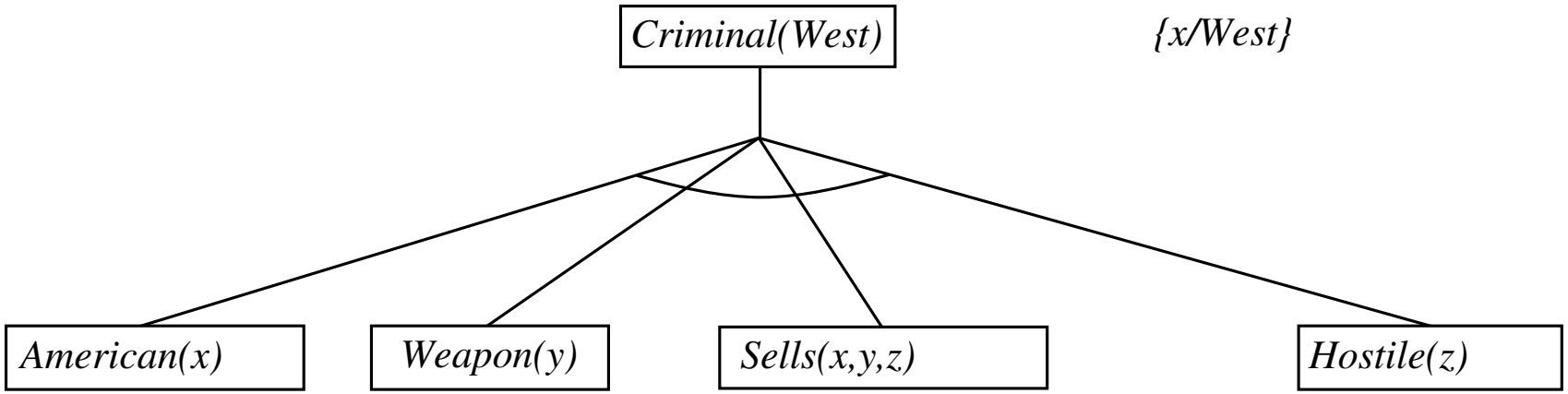
```
function FOL-BC-Ask(KB, goals, θ) returns a set of substitutions
    inputs:  KB, a knowledge base
             goals, a list of conjuncts forming a query (θ already applied)
             θ, the current substitution, initially the empty substitution { }
    local variables:  answers, a set of substitutions, initially empty

    if goals is empty then return {θ}
    q' ← Subst(θ, First(goals))
    for each sentence r in KB
                where Standardize-Apart(r) = (p1 ∧ ... ∧ pn ⇒ q)
                and θ' ← Unify(q, q') succeeds
          new_goals ← [p1, ..., pn | Rest(goals)]
          answers ← FOL-BC-Ask(KB, new_goals, Compose(θ', θ)) ∪ answers
    return answers
```

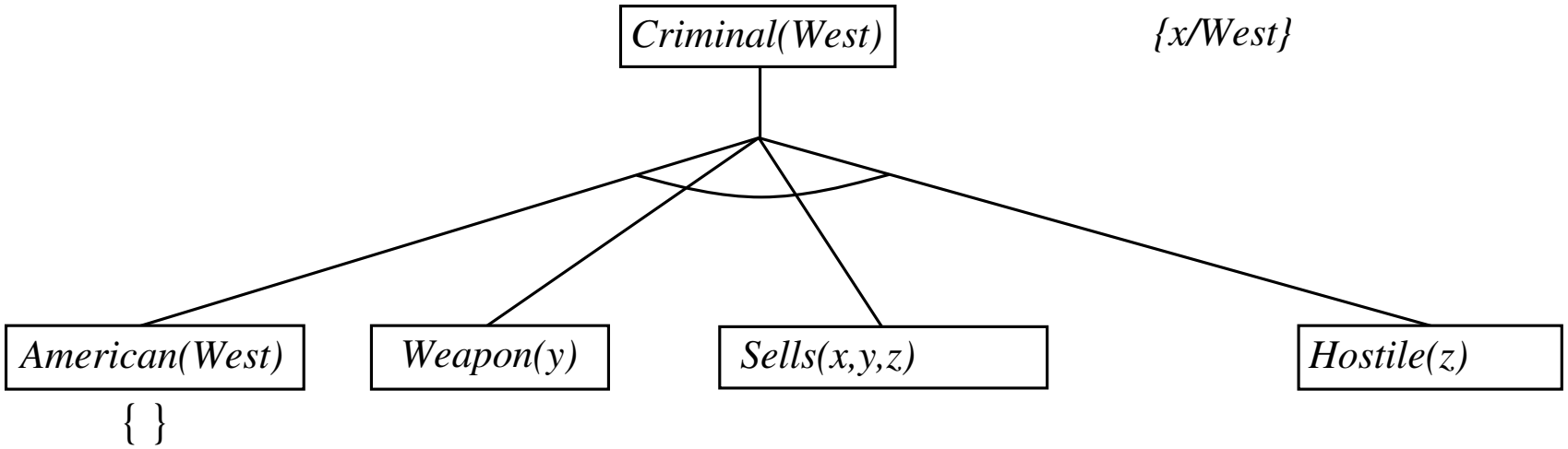
Backward chaining example

Criminal(West)

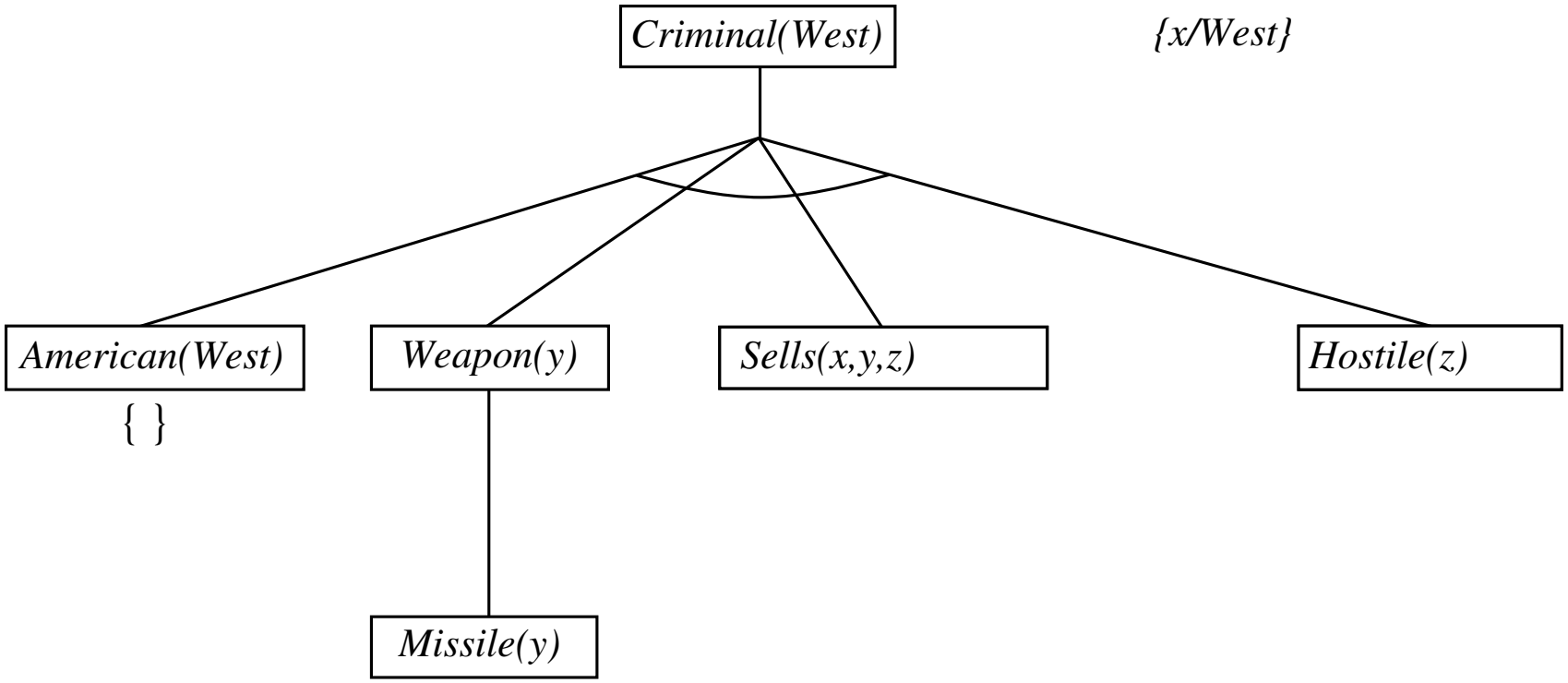
Backward chaining example



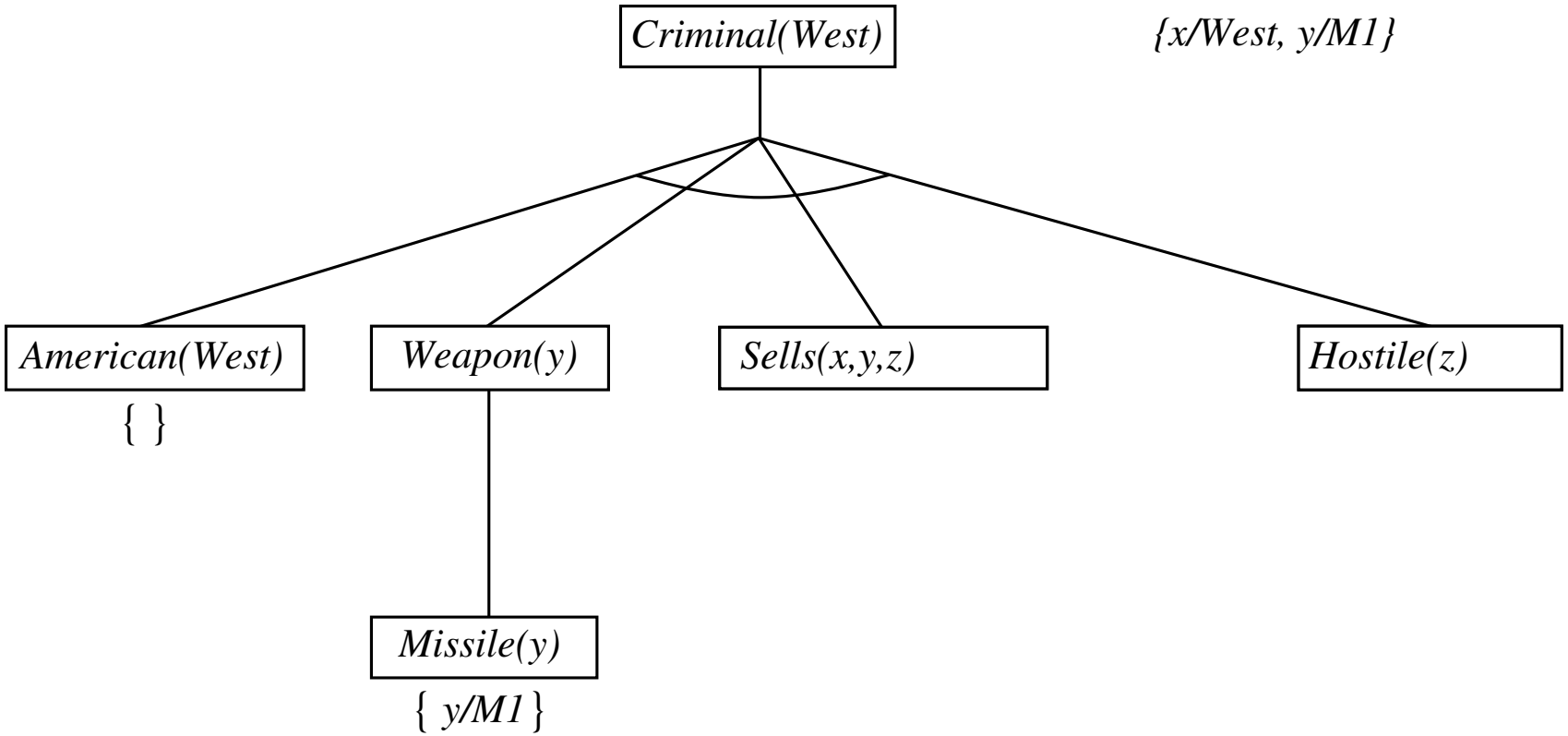
Backward chaining example



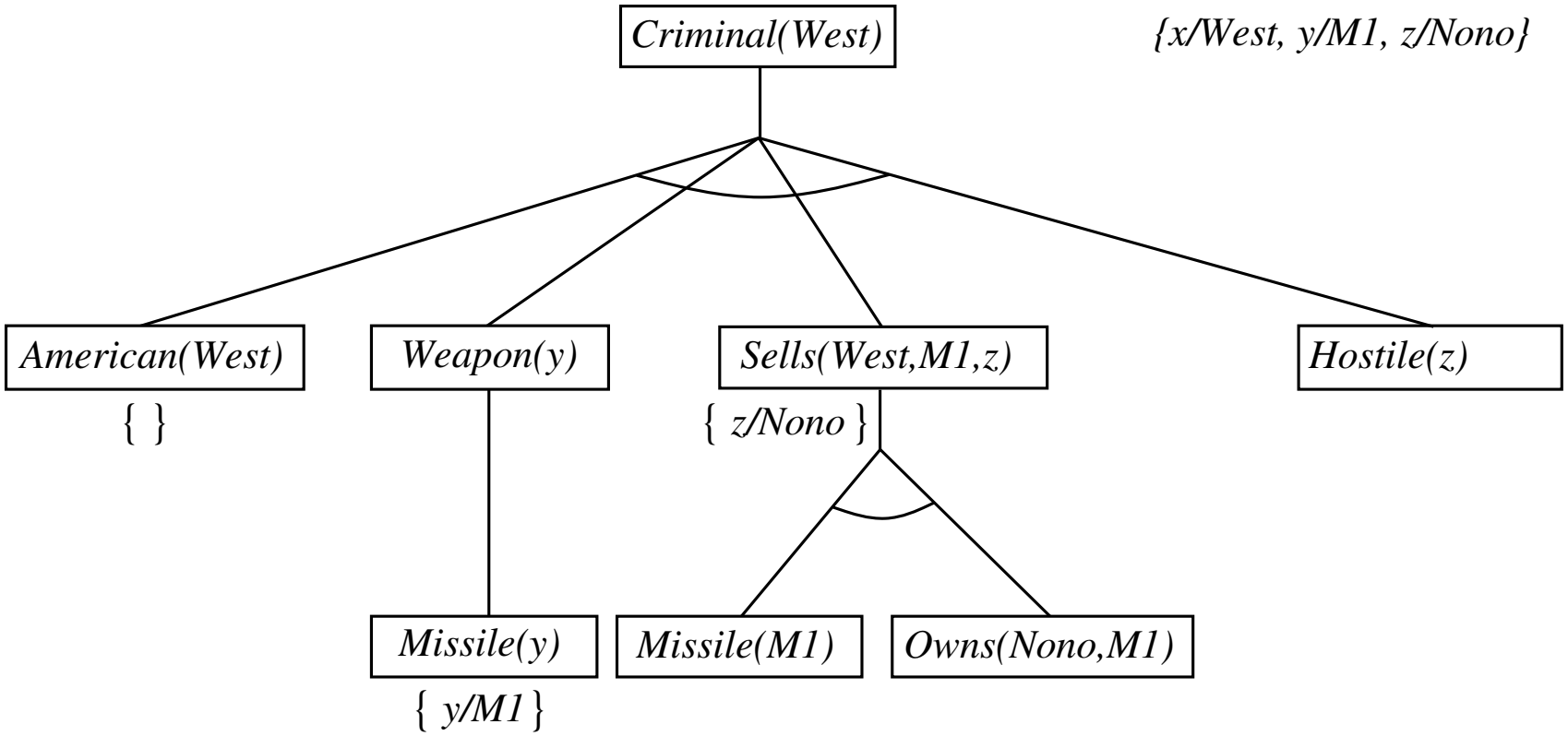
Backward chaining example



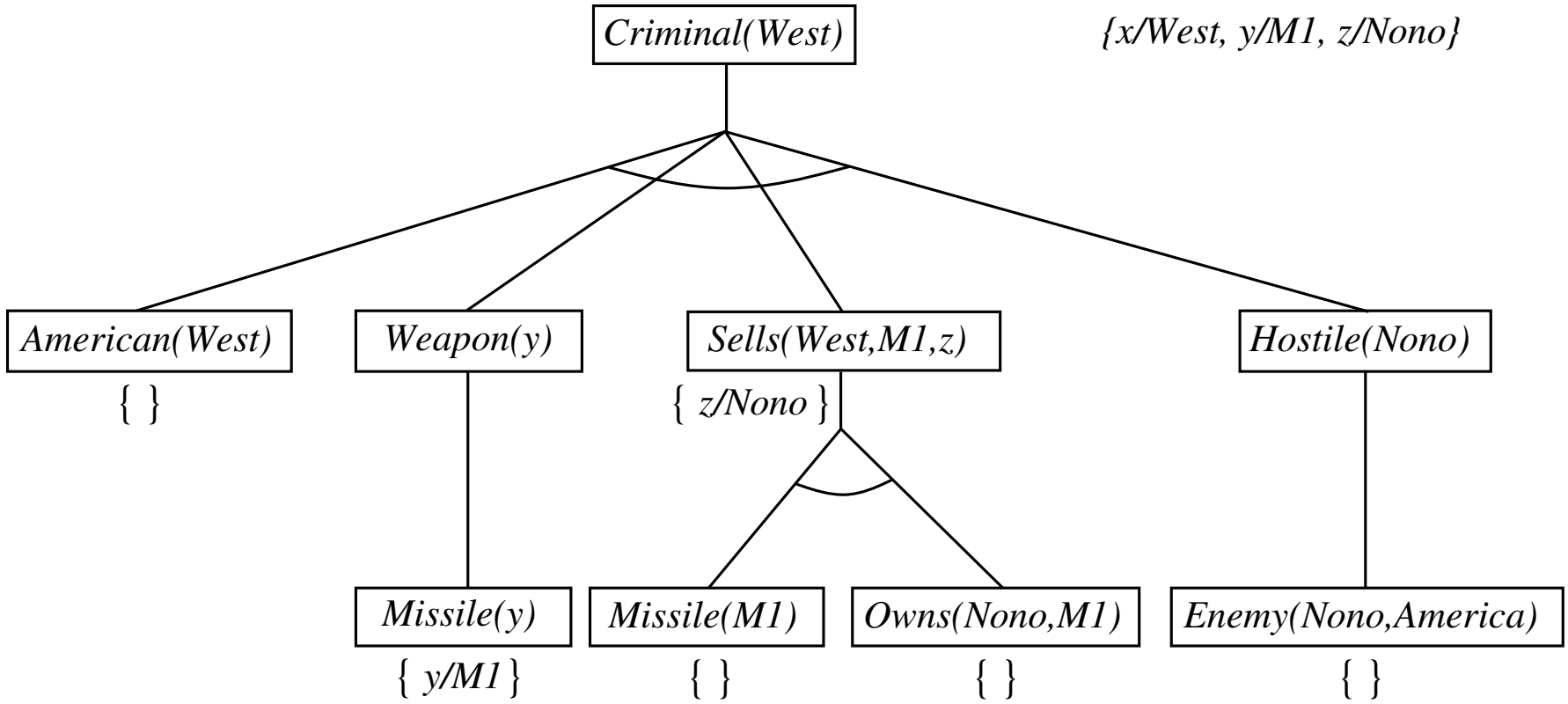
Backward chaining example



Backward chaining example



Backward chaining example



Properties of backward chaining

- Depth-first recursive proof search: space is linear in size of proof
- Incomplete due to infinite loops

  $\Rightarrow$ fix by checking current goal against every goal on stack
- Inefficient due to repeated subgoals (both success and failure)

  $\Rightarrow$ fix using caching of previous results (extra space!)
- Widely used (without improvements!) for logic programming

Logic programming

Algorithm = Logic + Control

Sound bite: computation as inference on logical KBs

	Logic programming	Ordinary programming
1.	Identify problem	Identify problem
2.	Assemble information	Assemble information
3.	Tea break	Figure out solution
4.	Encode information in KB	Program solution
5.	Encode problem instance as facts	Encode problem instance as data
6.	Ask queries	Apply program to data
7.	Find false facts	Debug procedural errors

Should be easier to debug $\mathsf{Capital}(\mathsf{NewYork}, \mathsf{US})$ than $x := x + 2$!

Prolog systems

- Basis: backward chaining with Horn clauses + bells & whistles
- Widely used in Europe, Japan (basis of 5th Generation project)
- Compilation techniques $\Rightarrow$ approaching a billion LIPS
- Program = set of clauses = `head :- literal`$_1$`, ... literal`$_n$`.`

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

- Efficient unification $\Longrightarrow$ omits occur-check!!
- 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`

Prolog examples

- Closed-world assumption (“negation as failure”)
  e.g., given `alive(X) :- not dead(X).`
  `alive(joe)` succeeds if `dead(joe)` fails
- 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

(Very Generalized Modus Ponens)

Resolution: brief summary

Full first-order version:

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

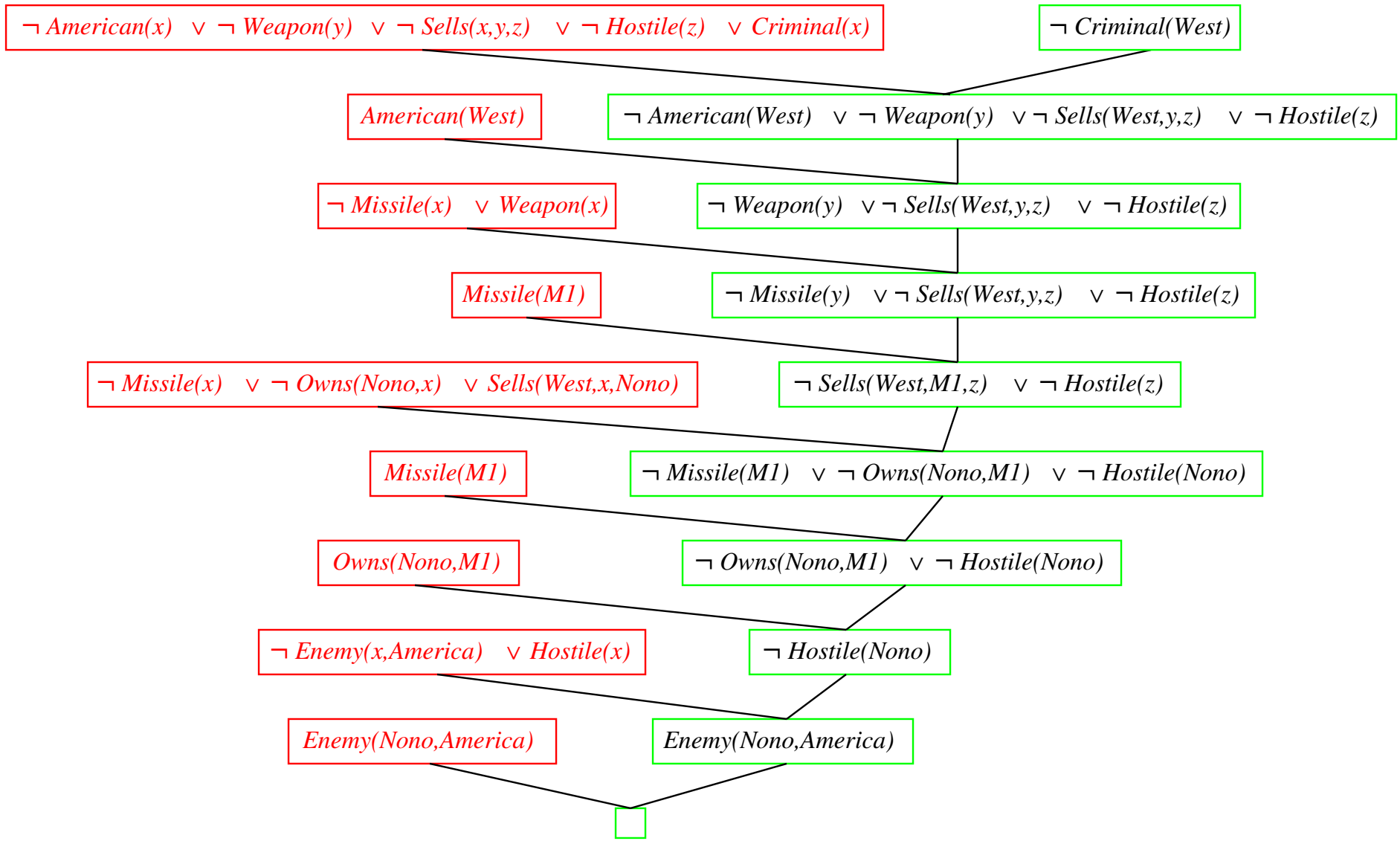
where $\textsc{Unify}(\ell_i, \neg m_j) = \theta$.

For example,

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

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

A Resolution Poof



Resolution: brief summary

[1] $P(a) \vee P(Y)$

[2] $\neg P(a) \vee \neg P(Y)$

Resolution: brief summary

[1] $P(a) \vee P(Y)$

[2] $\neg P(a) \vee \neg P(Y)$

[3] $P(Y) \vee \neg P(Y)$ by [Res 1.1&2.1]

Resolution: brief summary

[1] $P(a) \vee P(Y)$

[2] $\neg P(a) \vee \neg P(Y)$

[3] $P(Y) \vee \neg P(Y)$ by [Res 1.1&2.1]

[4] $P(Y) \vee \neg P(a)$ by [Res 1.1&2.2]

Resolution: brief summary

[1] $P(a) \vee P(Y)$

[2] $\neg P(a) \vee \neg P(Y)$

[3] $P(Y) \vee \neg P(Y)$ by [Res 1.1&2.1]

[4] $P(Y) \vee \neg P(a)$ by [Res 1.1&2.2]

[5] $P(a) \vee \neg P(Y)$ by [Res 1.2&2.1]

Resolution: brief summary

[1] $P(a) \vee P(Y)$

[2] $\neg P(a) \vee \neg P(Y)$

[3] $P(Y) \vee \neg P(Y)$ by [Res 1.1&2.1]

[4] $P(Y) \vee \neg P(a)$ by [Res 1.1&2.2]

[5] $P(a) \vee \neg P(Y)$ by [Res 1.2&2.1]

[6] $\neg P(a) \vee P(a)$ by [Res 1.2&2.2]

Resolution: brief summary

[1] $P(a) \vee P(Y)$

[2] $\neg P(a) \vee \neg P(Y)$

[3] $P(Y) \vee \neg P(Y)$ by [Res 1.1&2.1]

[4] $P(Y) \vee \neg P(a)$ by [Res 1.1&2.2]

[5] $P(a) \vee \neg P(Y)$ by [Res 1.2&2.1]

[6] $\neg P(a) \vee P(a)$ by [Res 1.2&2.2]

[7] $P(a) \vee P(Y)$ by [Res 1.1&3.2]

Resolution: brief summary

[1] $P(a) \vee P(Y)$

[2] $\neg P(a) \vee \neg P(Y)$

[3] $P(Y) \vee \neg P(Y)$ by [Res 1.1&2.1]

[4] $P(Y) \vee \neg P(a)$ by [Res 1.1&2.2]

[5] $P(a) \vee \neg P(Y)$ by [Res 1.2&2.1]

[6] $\neg P(a) \vee P(a)$ by [Res 1.2&2.2]

[7] $P(a) \vee P(Y)$ by [Res 1.1&3.2]

[8] $P(a) \vee \neg P(a)$ by [Res 1.2&3.2]

Resolution: brief summary

[1] $P(a) \vee P(Y)$

[2] $\neg P(a) \vee \neg P(Y)$

[3] $P(Y) \vee \neg P(Y)$ by [Res 1.1&2.1]

[4] $P(Y) \vee \neg P(a)$ by [Res 1.1&2.2]

[5] $P(a) \vee \neg P(Y)$ by [Res 1.2&2.1]

[6] $\neg P(a) \vee P(a)$ by [Res 1.2&2.2]

[7] $P(a) \vee P(Y)$ by [Res 1.1&3.2]

[8] $P(a) \vee \neg P(a)$ by [Res 1.2&3.2]

[7] $\neg P(a) \vee \neg P(Y)$ by [Res 2.1&3.1], [Res 2.2&3.1]

Resolution: brief summary

[1] $P(a) \vee P(Y)$

[2] $\neg P(a) \vee \neg P(Y)$

[3] $P(Y) \vee \neg P(Y)$ by [Res 1.1&2.1]

[4] $P(Y) \vee \neg P(a)$ by [Res 1.1&2.2]

[5] $P(a) \vee \neg P(Y)$ by [Res 1.2&2.1]

[6] $\neg P(a) \vee P(a)$ by [Res 1.2&2.2]

[7] $P(a) \vee P(Y)$ by [Res 1.1&3.2]

[8] $P(a) \vee \neg P(a)$ by [Res 1.2&3.2]

[7] $\neg P(a) \vee \neg P(Y)$ by [Res 2.1&3.1], [Res 2.2&3.1]

[9] $P(a) \vee \neg P(Y)$ by [Res 1.1&6.1],

Resolution: brief summary

[1] $P(a) \vee P(Y)$

[2] $\neg P(a) \vee \neg P(Y)$

[3] $P(Y) \vee \neg P(Y)$ by [Res 1.1&2.1]

[4] $P(Y) \vee \neg P(a)$ by [Res 1.1&2.2]

[5] $P(a) \vee \neg P(Y)$ by [Res 1.2&2.1]

[6] $\neg P(a) \vee P(a)$ by [Res 1.2&2.2]

[7] $P(a) \vee P(Y)$ by [Res 1.1&3.2]

[8] $P(a) \vee \neg P(a)$ by [Res 1.2&3.2]

[7] $\neg P(a) \vee \neg P(Y)$ by [Res 2.1&3.1], [Res 2.2&3.1]

[9] $P(a) \vee \neg P(Y)$ by [Res 1.1&6.1],

[10] $P(a) \vee P(a)$ by [Res 1.2&6.1],

Resolution: brief summary

[1] $P(a) \vee P(Y)$

[2] $\neg P(a) \vee \neg P(Y)$

[3] $P(Y) \vee \neg P(Y)$ by [Res 1.1&2.1]

[4] $P(Y) \vee \neg P(a)$ by [Res 1.1&2.2]

[5] $P(a) \vee \neg P(Y)$ by [Res 1.2&2.1]

[6] $\neg P(a) \vee P(a)$ by [Res 1.2&2.2]

[7] $P(a) \vee P(Y)$ by [Res 1.1&3.2]

[8] $P(a) \vee \neg P(a)$ by [Res 1.2&3.2]

[7] $\neg P(a) \vee \neg P(Y)$ by [Res 2.1&3.1], [Res 2.2&3.1]

[9] $P(a) \vee \neg P(Y)$ by [Res 1.1&6.1],

[10] $P(a) \vee P(a)$ by [Res 1.2&6.1],

$\vdots$

Resolution: brief summary

- Need a factorization rule for completeness like for propositional resolution
- **factorization** $\dfrac{l_1 \vee \ldots l_i \vee \ldots \vee l_j \vee \ldots l_n}{(l_1 \vee \ldots l_i \vee \ldots \vee l_{j-1} \vee l_{j+1} \vee \ldots l_n)\theta}$
  if θ is the mgu of l_i and l_j
- Resolution procedure:
  - Given a knowledge base and a conjecture α
  - Transform $KB \cup \{\neg\alpha\}$ into **clause normalform (CNF)**
  - Apply resolution and factorization rules to derive the empty clause
- Complete for FOL

Conversion to CNF

Everyone who loves all animals is loved by someone:

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

1. Eliminate biconditionals and implications

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

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

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

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

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

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:

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

5. Drop universal quantifiers:

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

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

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

Resolution Proof Example

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

$$\boxed{\neg\text{Loves}(x, F(x)) \vee \neg\text{Loves}(G(x), x)} \quad \boxed{\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)}$$

Resolution Proof Example

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

$$\boxed{\neg\text{Loves}(x, F(x)) \vee \neg\text{Loves}(G(x), x)} \quad \boxed{\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)}$$

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

$$\boxed{\neg\text{Animal}(z) \vee \neg\text{Kills}(x, z) \vee \neg\text{Loves}(y, x)}$$

Resolution Proof Example

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

  $\boxed{\neg\text{Loves}(x, F(x)) \vee \neg\text{Loves}(G(x), x)}$ $\boxed{\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)}$

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

  $\boxed{\neg\text{Animal}(z) \vee \neg\text{Kills}(x, z) \vee \neg\text{Loves}(y, x)}$

- $\forall x\ \text{Animal}(x) \Rightarrow \text{Loves}(\text{Jack}, x)$ $\boxed{\neg\text{Animal}(x) \vee \text{Loves}(\text{Jack}, x)}$

Resolution Proof Example

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

$$\boxed{\neg\text{Loves}(x, F(x)) \vee \neg\text{Loves}(G(x), x)} \quad \boxed{\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)}$$

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

$$\boxed{\neg\text{Animal}(z) \vee \neg\text{Kills}(x, z) \vee \neg\text{Loves}(y, x)}$$

- $\forall x\ \text{Animal}(x) \Rightarrow \text{Loves}(\text{Jack}, x)$ $\quad \boxed{\neg\text{Animal}(x) \vee \text{Loves}(\text{Jack}, x)}$

- $\text{Kills}(\text{Jack}, \text{Tuna}) \vee \text{Kills}(\text{Curiosity}, \text{Tuna})$

$$\boxed{\text{Kills}(\text{Jack}, \text{Tuna}) \vee \text{Kills}(\text{Curiosity}, \text{Tuna})}$$

Resolution Proof Example

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

  $\boxed{\neg\text{Loves}(x, F(x)) \vee \neg\text{Loves}(G(x), x)}$ $\boxed{\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)}$

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

  $\boxed{\neg\text{Animal}(z) \vee \neg\text{Kills}(x, z) \vee \neg\text{Loves}(y, x)}$

- $\forall x\ \text{Animal}(x) \Rightarrow \text{Loves}(\text{Jack}, x)$ $\boxed{\neg\text{Animal}(x) \vee \text{Loves}(\text{Jack}, x)}$

- $\text{Kills}(\text{Jack}, \text{Tuna}) \vee \text{Kills}(\text{Curiosity}, \text{Tuna})$

  $\boxed{\text{Kills}(\text{Jack}, \text{Tuna}) \vee \text{Kills}(\text{Curiosity}, \text{Tuna})}$

- $\text{Cat}(\text{Tuna})$ $\boxed{\text{Cat}(\text{Tuna})}$

Resolution Proof Example

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

  $\boxed{\neg\text{Loves}(x,F(x)) \vee \neg\text{Loves}(G(x),x)}$ $\boxed{\text{Animal}(F(x)) \vee \text{Loves}(G(x),x)}$

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

  $\boxed{\neg\text{Animal}(z) \vee \neg\text{Kills}(x,z) \vee \neg\text{Loves}(y,x)}$

- $\forall x\ \text{Animal}(x) \Rightarrow \text{Loves}(\text{Jack},x)$ $\boxed{\neg\text{Animal}(x) \vee \text{Loves}(\text{Jack},x)}$

- $\text{Kills}(\text{Jack},\text{Tuna}) \vee \text{Kills}(\text{Curiosity},\text{Tuna})$

  $\boxed{\text{Kills}(\text{Jack},\text{Tuna}) \vee \text{Kills}(\text{Curiosity},\text{Tuna})}$

- $\text{Cat}(\text{Tuna})$ $\boxed{\text{Cat}(\text{Tuna})}$

- $\forall x\ \text{Cat}(x) \Rightarrow \text{Animal}(x)$ $\boxed{\neg\text{Cat}(x) \vee \text{Animal}(x)}$

Resolution Proof Example

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

  $\boxed{\neg\text{Loves}(x,F(x)) \vee \neg\text{Loves}(G(x),x)}$ $\boxed{\text{Animal}(F(x)) \vee \text{Loves}(G(x),x)}$

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

  $\boxed{\neg\text{Animal}(z) \vee \neg\text{Kills}(x,z) \vee \neg\text{Loves}(y,x)}$

- $\forall x\ \text{Animal}(x) \Rightarrow \text{Loves}(\text{Jack},x)$ $\boxed{\neg\text{Animal}(x) \vee \text{Loves}(\text{Jack},x)}$

- $\text{Kills}(\text{Jack},\text{Tuna}) \vee \text{Kills}(\text{Curiosity},\text{Tuna})$

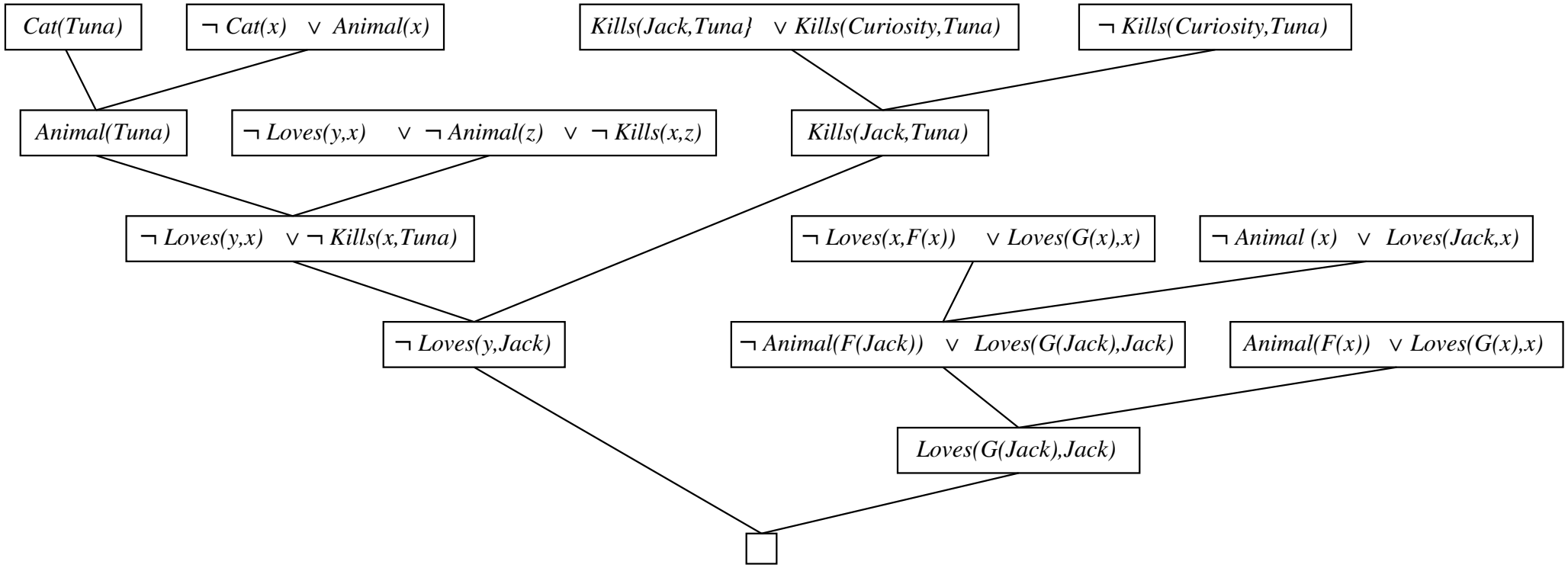
  $\boxed{\text{Kills}(\text{Jack},\text{Tuna}) \vee \text{Kills}(\text{Curiosity},\text{Tuna})}$

- $\text{Cat}(\text{Tuna})$ $\boxed{\text{Cat}(\text{Tuna})}$

- $\forall x\ \text{Cat}(x) \Rightarrow \text{Animal}(x)$ $\boxed{\neg\text{Cat}(x) \vee \text{Animal}(x)}$

- Negated conjecture: $\neg\text{Kills}(\text{Curiosity},\text{Tuna})$ $\boxed{\neg\text{Kills}(\text{Curiosity},\text{Tuna})}$

Resolution proof



- Use of factorization to derive Loves(G(Jack), Jack)