CS3243: Introduction to Artificial Intelligence

Semester 2, 2020

Inference in First-Order Logic (FOL)

AIMA Chapter 9.1 – 9.3, 9.4.1, 9.5.1 – 9.5.3

Outline

- Reduction to propositional inference
- Unification
- Generalized Modus Ponens
- Forward chaining
- Backward chaining
- Resolution

Converting from First Order to Propositional Logic

- Universal quantifier ⇒ propositional rules:
  $\forall x{:}\, P(x) \wedge Q(x) \Rightarrow R(x)$ becomes
  $$P(a) \wedge Q(a) \Rightarrow R(a)$$
  $$P(b) \wedge Q(b) \Rightarrow R(b)$$
  $$\vdots$$
- Convert an existential quantifier by adding a new **Skolem constant** (not appearing in KB!)
  $\exists x{:}\, P(x)$ becomes $P(x_0)$; the variable x_0 is **new**.

First Order to Propositional Logic

Rules:
$\mathrm{King}(John)$,
$\mathrm{Greedy}(John)$,
$\mathrm{Brother}(Richard, John)$,
$\forall x: \mathrm{King}(x) \land \mathrm{Greedy}(x) \Rightarrow \mathrm{Evil}(x)$,
$\exists x: \mathrm{Crown}(x) \land \mathrm{Onhead}(x, John)$.

Rules:
$\mathrm{King}(John)$,
$\mathrm{Greedy}(John)$,
$\mathrm{Brother}(Richard, John)$,
$\mathrm{King}(John) \land \mathrm{Greedy}(John) \Rightarrow \mathrm{Evil}(John)$,
$\mathrm{King}(Richard) \land \mathrm{Greedy}(Richard) \Rightarrow \mathrm{Evil}(Richard)$,
$\mathrm{Crown}(C) \land \mathrm{Onhead}(C, John)$.

Atomic sentences (e.g. $\mathrm{King}(John)$, $\mathrm{Brother}(Richard, John)$) are now symbols

Reduction to Propositional Inference

- Every FOL KB can be propositionalized; preserves entailment: α is entailed by new KB iff entailed by original KB
- **Idea:** propositionalize KB and query; infer, return result
- **Problem:** with function symbols, there are infinitely many ground-term substitutions
  - e.g., $x > y \Leftrightarrow \big(x = next(y)\big) \vee \Big(\exists z: \big(x = next(z)\big) \wedge (z > y)\Big)$
  - $x > y$ would convert to
    $x = next(y) \vee x = next\big(next(y)\big) \vee x = next\Big(next\big(next(y)\big)\Big)$ ...

Theorem (Herbrand, 1930). If α is entailed by FOL KB, then it is entailed by a **finite subset** of the propositionalized KB.

Does this remind you of any other algorithm?

- Idea: For $n = 0$ to ∞ do
  - create a propositionalized KB_n by instantiating with depth-n terms
  - see if α is entailed by this KB_n

What happens if α is not entailed?

Entailment for FOL is **semi-decidable** ($\exists$ algorithms that return TRUE if α is entailed, but no algorithm exists that returns FALSE to every non-entailed α).

Similar to the **halting problem (Turing, 1936)**

Propositionalization is Expensive

- Generates irrelevant things:

  $$\forall x\colon \mathrm{King}(x) \wedge \mathrm{Greedy}(x) \Rightarrow \mathrm{Evil}(x)$$
  $$\forall y\colon \mathrm{Greedy}(y)$$
  $$\mathrm{King}(John)$$

  Easily implies that $\mathrm{Evil}(John)$

  … but we will also add the irrelevant $\mathrm{Greedy}(Richard)$

- Exponential blowup: a k-ary predicate has d^k instantiations with d constants. We don't need all of them!

Key Idea - Unification

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

  Unifier $\theta = \{x \leftarrow John, y \leftarrow John\}$ works

  AIMA uses notation $x \backslash John$

- $\text{UNIF}Y(p, q)$ outputs a var. substitution θ s.t. $\text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)$

p	q	θ
$\text{Knows}(John, x)$	$\text{Knows}(John, Jane)$	$\{x \leftarrow Jane\}$
$\text{Knows}(John, x)$	$\text{Knows}(y, Bob)$	$\{x \leftarrow Bob, y \leftarrow John\}$
$Knows(John, x)$	$\text{Knows}(y, \text{Mother}(y))$	$\{y \leftarrow John, x \leftarrow \text{Mother}(John)\}$
$Knows(John, x)$	$Knows(x, Bob)$	FAIL

- **Standardizing apart** eliminates variable name clashes

Unification – Multiple Unifiers

To unify $\text{Knows}(John, x)$ and $\text{Knows}(y, z)$,

$\theta = \{y \leftarrow John, x \leftarrow z\}$ ⇐ **More General**

or

$\theta = \{y \leftarrow John, x \leftarrow John, z \leftarrow John\}$

There is a unique most general unifier (MGU), up to renaming and substitution of variables

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

Finding an MGU: 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

---

function UNIFY-VAR(var, x, θ) **returns** a substitution

  if $\{var/val\} \in \theta$ **then return** UNIFY(val, x, θ)
  else if $\{x/val\} \in \theta$ **then return** UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) **then return** failure
  else return add $\{var/x\}$ to θ

Unification Algorithm

Given two sentences P, Q, find a variable assignment such that $P = Q$

- Input: P, Q, θ (θ is empty at first)
- If $\theta = \text{FAIL}$ or if $P = Q$ return θ
- If one input is a variable (say, $P = x$), unify a variable with a sentence given θ.
- Otherwise, break up the compound function/list and process their parts recursively.

The UNIFY-VAR function

Given a variable x, a sentence Q, and θ

- If $x \leftarrow val$ (under θ), return $\mathrm{UNIFY}(val, Q, \theta)$; if $Q \leftarrow val$ (under θ), return $\mathrm{UNIFY}(x, val, \theta)$.
- If x appears in Q return FAIL (recursion won't stop!)
- If x does not appear in Q return $\theta \cup \{x \leftarrow Q\}$

It is ok to set x to be a very long sentence:
$x \leftarrow P(Q(y, z, \ell), A)$

Unification Example

Input: $F(x, y), F(a, z)$

$\text{Unify}(F(x, y), F(a, z), [,])$

$\text{Unify}\big([x, y], [a, z], \text{Unify}(F, F, [,])\big)$

$\text{Unify}([x, y], [a, z], [,])$

$\text{Unify}\big(x, a, \text{Unify}(y, z, [,])\big)$

$\text{Unify}(x, a, [y \leftarrow z]) \Rightarrow$
$[y \leftarrow z, x \leftarrow a]$

Inference Algorithms for FOL

Generalized Modus Ponens (GMP)

- Original Modus Ponens: $\frac{\alpha_1, \ldots, \alpha_k; (\alpha_1 \wedge \cdots \wedge \alpha_k \Rightarrow \beta)}{\beta}$
- Generalized (lifted) Modus Ponens:
  - $\mathrm{King}(John), \forall y\colon \mathrm{Greedy}(y)$
  - $\forall x\colon King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
  - **There is some substitution $\boldsymbol{\theta}$** such that the premise and the implication are the same (namely $[x \leftarrow John, y \leftarrow John]$)
  - … we can infer that $Evil(John)$

Generalized Modus Ponens (GMP)

These sentences have variables $x_1, \dots, x_n$ in their universal quantifiers (the ∀ elements)

- We are given a set of FOL sentences $P_1, \dots, P_k$ and an implication rule $R_1 \wedge \cdots \wedge R_k \Rightarrow Q$
- If there exists some substitution θ over the variables such that

$$\mathrm{SUBST}(P_1, \theta) = \mathrm{SUBST}(R_1, \theta)$$
$$\mathrm{SUBST}(P_2, \theta) = \mathrm{SUBST}(R_2, \theta)$$
$$\vdots$$
$$\mathrm{SUBST}(P_k, \theta) = \mathrm{SUBST}(R_k, \theta)$$

- … then $\mathrm{SUBST}(Q; \theta)$ holds!

Soundness of GMP

Assume:

- there exists a substitution θ such that $\text{SUBST}(P_j, \theta) = \text{SUBST}(R_j, \theta)$ for all j.
- $P_1, \dots, P_k$ hold, and that
- $R_1 \wedge \cdots \wedge R_k \Rightarrow Q$

We need to prove that $\text{SUBST}(Q, \theta)$ holds as well.

$\exists\theta$ s.t. $\mathrm{SUBST}(P_j, \theta) = \mathrm{SUBST}(R_j, \theta)\ \forall j$.

$P_1, \dots, P_k$ hold, and $R_1 \wedge \cdots \wedge R_k \Rightarrow Q$

Lemma: if $P = \forall y_1, \dots, y_\ell: \alpha(y_1, \dots, y_\ell)$ then $P \vDash \mathrm{SUBST}(P, \theta)$ for any substitution θ

Proof: by universal instantiation

- By Lemma:
  - $\forall j: P_j \vDash \mathrm{SUBST}(P_j, \theta) = \mathrm{SUBST}(R_j, \theta)$ and
  - $R_1 \wedge \cdots \wedge R_k \Rightarrow Q \vDash$
    $\mathrm{SUBST}(R_1, \theta) \wedge \cdots \wedge \mathrm{SUBST}(R_k, \theta) \Rightarrow \mathrm{SUBST}(Q, \theta)$
- We know that
  - $\mathrm{SUBST}(R_1, \theta), \dots, \mathrm{SUBST}(R_k, \theta)$ hold, and that
  - $\mathrm{SUBST}(R_1, \theta) \wedge \cdots \wedge \mathrm{SUBST}(R_k, \theta) \Rightarrow \mathrm{SUBST}(Q, \theta)$
- Apply Propositional Logic's Modus Ponens!

Example KB in FOL

"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."

You may assume that

1. Missiles are weapons
2. Any country that's an enemy of America is a hostile nation

Prove that Colonel West is a criminal…
Using your FOL superpowers

Example KB in FOL

"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."

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

Example KB in FOL

"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."

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

Example KB in FOL

“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.”

$$\exists x\text{: Owns}(Nono, x) \wedge \text{Missile}(x)$$

Existential Instantiation

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

Example KB in FOL

"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."

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

Example KB in FOL

"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**."

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

Example KB in FOL

- Missiles are weapons:

$$\forall x\colon \mathrm{Missile}(x) \Rightarrow \mathrm{Weapon}(x)$$

- Enemies of America are hostile:

$$\forall x\colon \mathrm{Enemy}(x, America) \Rightarrow \mathrm{Hostile}(x)$$

Example KB in FOL

1. $\forall x, y, z\colon \text{American}(x) \land \text{Weapon}(y) \land \text{Hostile}(z) \land \text{Sells}(x, y, z) \Rightarrow \text{Criminal}(x)$
2. $\text{Enemy}(Nono, America)$
3. $\text{Owns}(Nono, M_1)$
4. $\text{Missile}(M_1)$
5. $\forall x\colon \text{Missile}(x) \land \text{Owns}(Nono, x) \Rightarrow \text{Sells}(West, x, Nono)$
6. $\text{American}(West)$
7. $\forall x\colon \text{Missile}(x) \Rightarrow \text{Weapon}(x)$
8. $\forall x\colon \text{Enemy}(x, America) \Rightarrow \text{Hostile}(x)$

Example KB in FOL

Want to prove that

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

Forward Chaining Algorithm

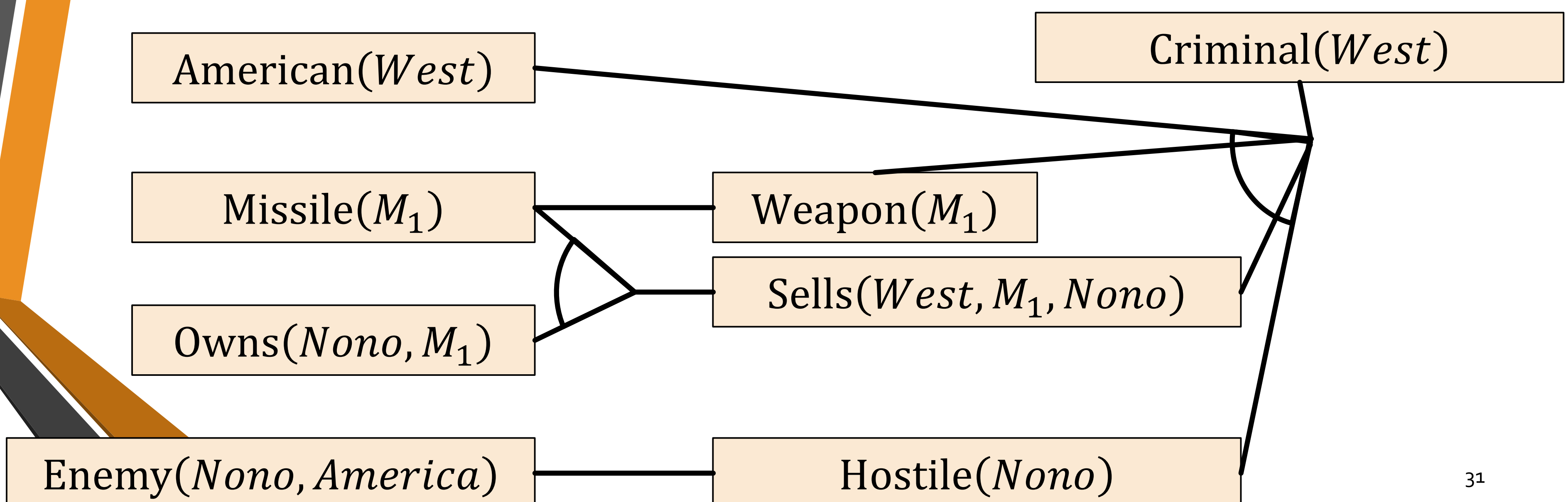
```
function FOL-FC-ASK(KB, α) returns a substitution or false
  inputs: KB, the knowledge base, a set of first-order definite clauses
          α, the query, an atomic sentence
  local variables: new, the new sentences inferred on each iteration

  repeat until new is empty
      new ← { }
      for each rule in KB do
          ( p1 ∧ . . . ∧ pn ⇒ q) ← STANDARDIZE-VARIABLES(rule)
          for each θ such that SUBST(θ, p1 ∧ . . . ∧ pn) = SUBST(θ, p'1 ∧ . . . ∧ p'n)
                        for some p'1, . . . , p'n in KB
              q' ← SUBST(θ, q)
              if q' does not unify with some sentence already in KB or new then
                  add q' to new
                  φ ← UNIFY(q', α)
                  if φ is not fail then return φ
      add new to KB
  return false
```

Forward Chaining Algorithm

- Input:
  - KB – a knowledge base coded in FOL
  - α – an FOL sentence
- Output:
  - A variable substitution θ so that $SUBST(KB, \theta) \vDash \alpha$
  - … or FALSE if no such θ exists.

- At every round, add all newly inferred atomic sentences to KB.
- Repeat until
  - One of these sentences is α
  - No new sentences can be inferred.

1. $\forall x, y, z: \text{American}(x) \land \text{Weapon}(y) \land \text{Hostile}(z) \land \text{Sells}(x, y, z) \Rightarrow \text{Criminal}(x)$
2. $\text{Enemy}(Nono, America)$
3. $\text{Owns}(Nono, M_1)$
4. $\text{Missile}(M_1)$
5. $\forall x: \text{Missile}(x) \land \text{Owns}(Nono, x) \Rightarrow \text{Sells}(West, x, Nono)$
6. $\text{American}(West)$
7. $\forall x: \text{Missile}(x) \Rightarrow \text{Weapon}(x)$
8. $\forall x: \text{Enemy}(x, America) \Rightarrow \text{Hostile}(x)$



Properties of Forward Chaining

- Sound and complete for first-order definite clauses
- FC terminates for KB in finite number of iterations if it contains no functions.
- May not terminate in general (i.e., with functions) if α is not entailed
- This is unavoidable: entailment with definite clauses is semidecidable

Inefficiencies of Forward Chaining

function FOL-FC-ASK(KB, α) **returns** a substitution or *false*
  inputs: KB, the knowledge base, a set of first-order definite clauses
        α, the query, an atomic sentence
  local variables: new, the new sentences inferred on each iteration

  repeat until new is empty
    $new \leftarrow \{\}$
    for each $rule$ in KB **do**
      $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-VARIABLES($rule$)
      for each θ such that SUBST($\theta, p_1 \wedge \ldots \wedge p_n$) = SUBST($\theta, p'_1 \wedge \ldots \wedge p'_n$)
            for some $p'_1, \ldots, p'_n$ in KB
        $q' \leftarrow$ SUBST(θ, q)
        if q' does not unify with some sentence already in KB or new **then**
          add q' to new
          $\phi \leftarrow$ UNIFY(q', α)
          if ϕ is not *fail* **then return** ϕ
    add new to KB
  return *false*

Matching rules and known facts is costly

Matching Rule Premises to Known Facts is Costly

Predicate indexing: constant time to retrieve known facts

- $\text{Missile}(x) \Rightarrow \text{Weapon}(x)$: find all facts (e.g., $\text{Missile}(M_1)$) that unify with $\text{Missile}(x)$.

Conjunct ordering problem: apply minimum-remaining-values heuristic of CSP

- e.g., $\text{Missile}(x) \wedge \text{Owns}(Nono, x) \Rightarrow \text{Sells}(West, x, Nono)$
  if many objects are owned by $Nono$ and few missiles, start with $\text{Missile}(x)$ conjunct

Inefficiencies of Forward Chaining

function FOL-FC-ASK(KB, α) **returns** a substitution or *false*

inputs: KB, the knowledge base, a set of first-order definite clauses

α, the query, an atomic sentence

local variables: new, the new sentences inferred on each iteration

repeat until new is empty

$new \leftarrow \{\}$

for each $rule$ **in** KB **do**

$(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-VARIABLES($rule$)

for each θ such that SUBST($\theta, p_1 \wedge \ldots \wedge p_n$) = SUBST($\theta, p'_1 \wedge \ldots \wedge p'_n$)

for some $p'_1, \ldots, p'_n$ in KB

$q' \leftarrow$ SUBST(θ, q)

if q' does not unify with some sentence already in KB or new **then**

add q' to new

$\phi \leftarrow$ UNIFY(q', α)

if ϕ is not *fail* **then return** ϕ

add new to KB

return *false*

Redundant rule matching

Redundant Rule Matchings

Incremental forward chaining: Match rule at time t only if a conjunct in its premise unifies with **new fact** inferred at iteration t– 1.

- e.g., $\text{Missile}(x) \Rightarrow \text{Weapon}(x)$ matches against $\text{Missile}(M_1)$ again in 2nd iteration; but, $\text{Weapon}(M_1)$ is already known.

Rete ("Ree-Tee") algorithm:

- Don't discard partially matched rules
- Keep track of conjuncts matched against new facts; avoid duplicate work: $\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$ is partially matched against $\text{American}(West)$ in first iteration.

Inefficiencies of Forward Chaining

function FOL-FC-ASK(KB, α) **returns** a substitution or *false*
  inputs: KB, the knowledge base, a set of first-order definite clauses
    α, the query, an atomic sentence
  local variables: new, the new sentences inferred on each iteration

  repeat until new is empty
    $new \leftarrow \{\ \}$
    for each $rule$ in KB **do**
      $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-VARIABLES($rule$)
      for each θ such that SUBST($\theta, p_1 \wedge \ldots \wedge p_n$) = SUBST($\theta, p'_1 \wedge \ldots \wedge p'_n$)
          for some $p'_1, \ldots, p'_n$ in KB
        $q' \leftarrow$ SUBST(θ, q)
        if q' does not unify with some sentence already in KB or new **then**
          add q' to new
          $\phi \leftarrow$ UNIFY(q', α)
          if ϕ is not $fail$ **then return** ϕ
    add new to KB
  return $false$

Generating Irrelevant Facts

Backward Chaining Algorithm

function FOL-BC-ASK(K
  return FOL-BC-OR(K

At least one of the rules in the KB needs to imply $goal$

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) ← STANDARDIZE-VARIABLES((lhs, rhs))
    for each θ' **in** FOL-B
      yield θ'

All elements of $goals$ need to be true

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

Backward Chaining: Proof Tree

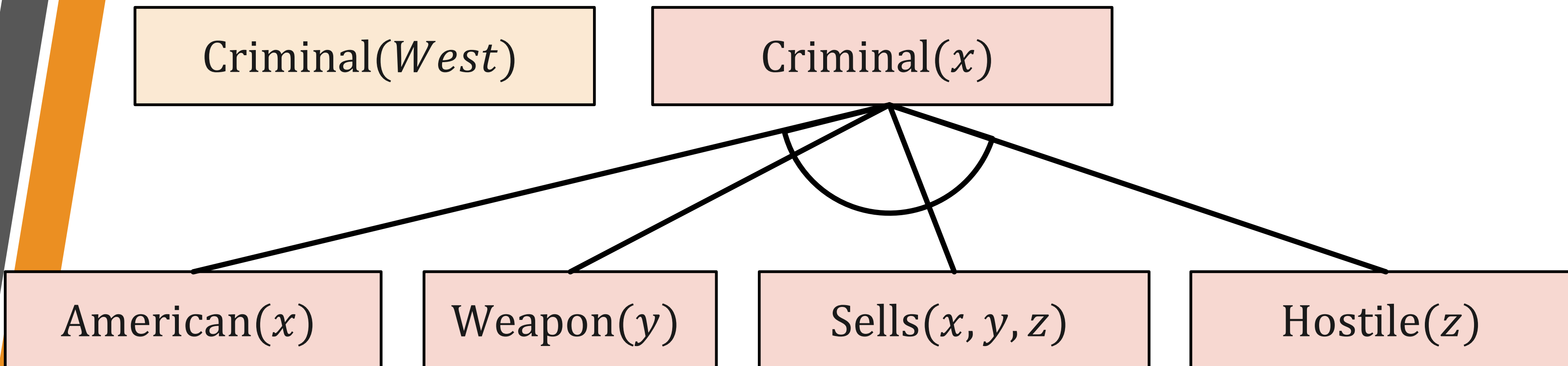
$\text{Criminal}(West)$

Call FOL-BC-OR on KB and $\text{Criminal}(West)$

FETCH-RULE on $\text{Criminal}(West)$:

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

Backward Chaining: Proof Tree

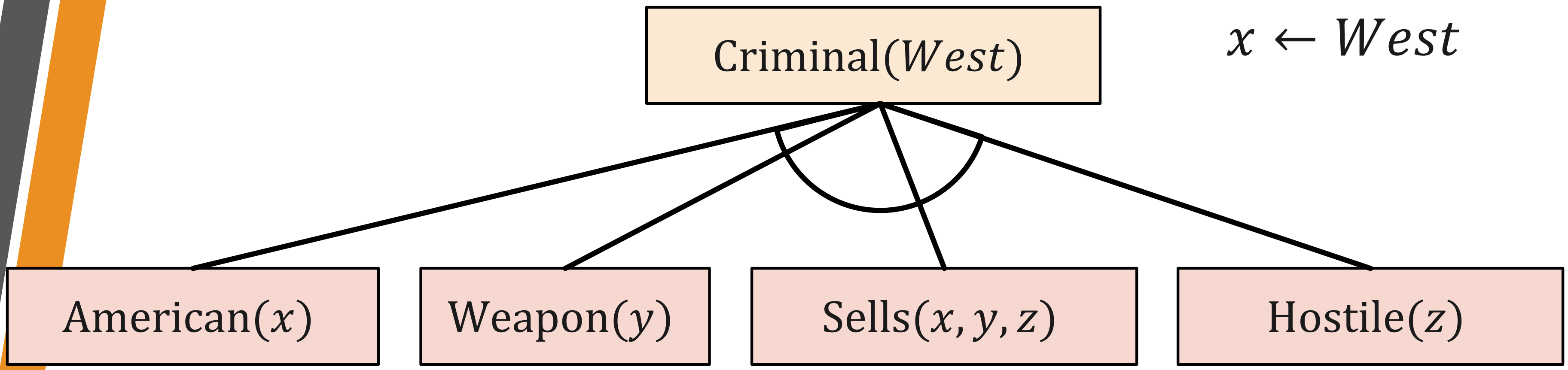


Unify

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

and $Criminal(West)$, result in $\theta = [x \leftarrow West]$

Backward Chaining: Proof Tree

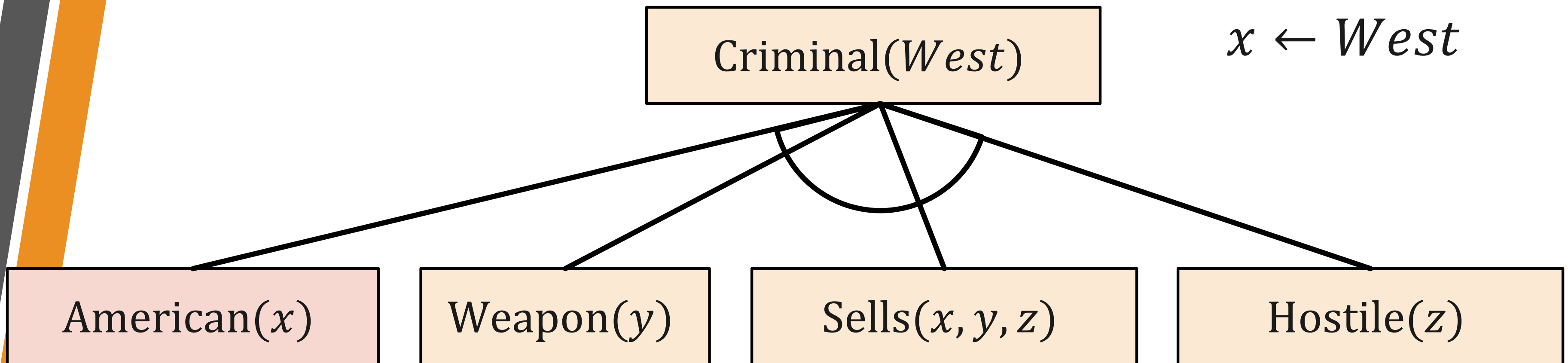


$x \leftarrow West$

Call FOL-BC-AND on the list of rules

$[\mathrm{American}(x), \mathrm{Weapon}(y), \mathrm{Sells}(x, y, z), \mathrm{Hostile}(z)]$

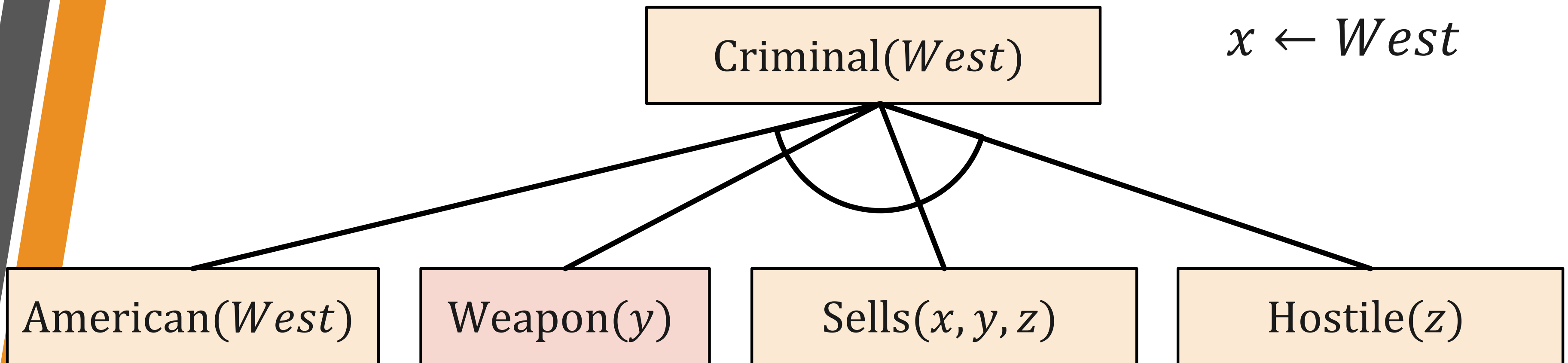
Backward Chaining: Proof Tree



American(x) gets unified with American($West$) in the KB with existing substitution

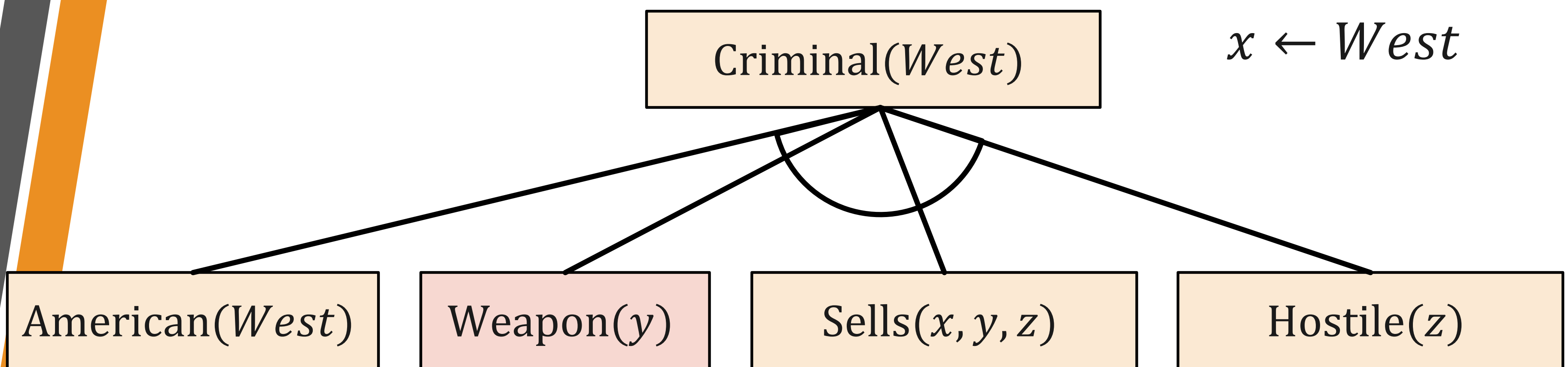
Backward Chaining: Proof Tree

$x \leftarrow West$



$\text{American}(x)$ gets unified with $\text{American}(West)$ in the KB with existing substitution

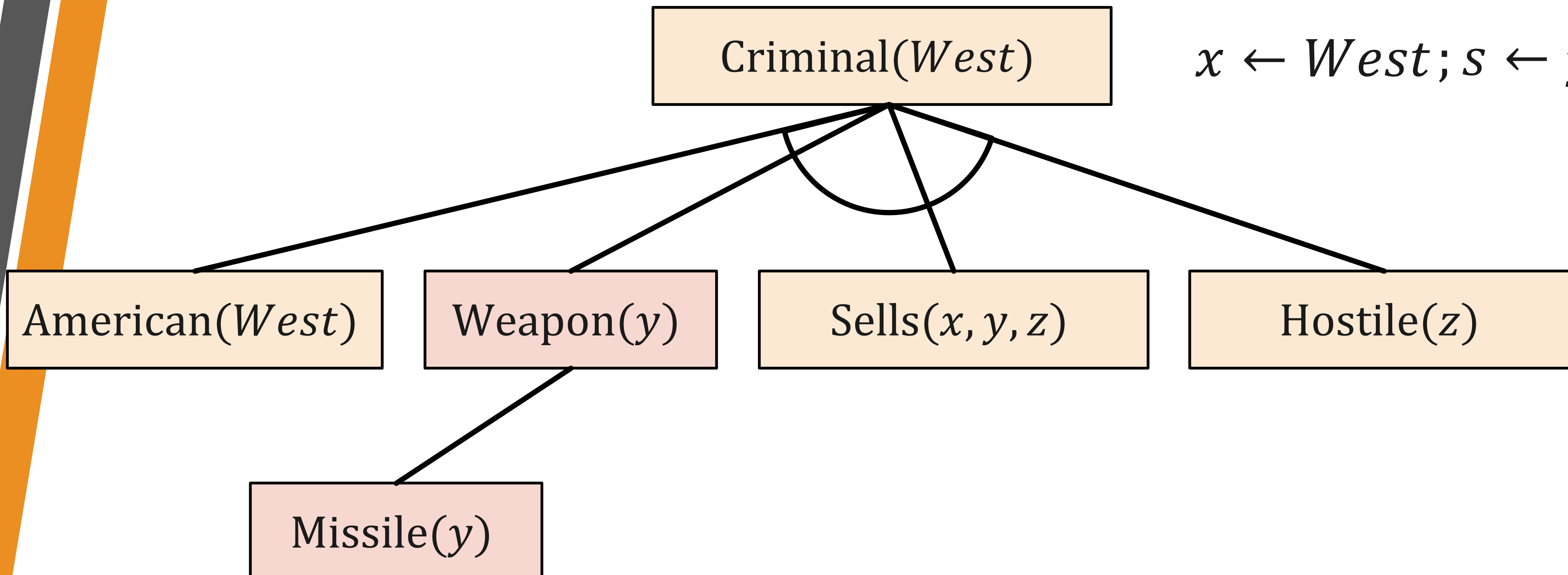
Backward Chaining: Proof Tree



We call FOL-BC-OR on $\text{Weapon}(y)$ and fetch the rule $\text{Missile}(s) \Rightarrow \text{Weapon}(s)$

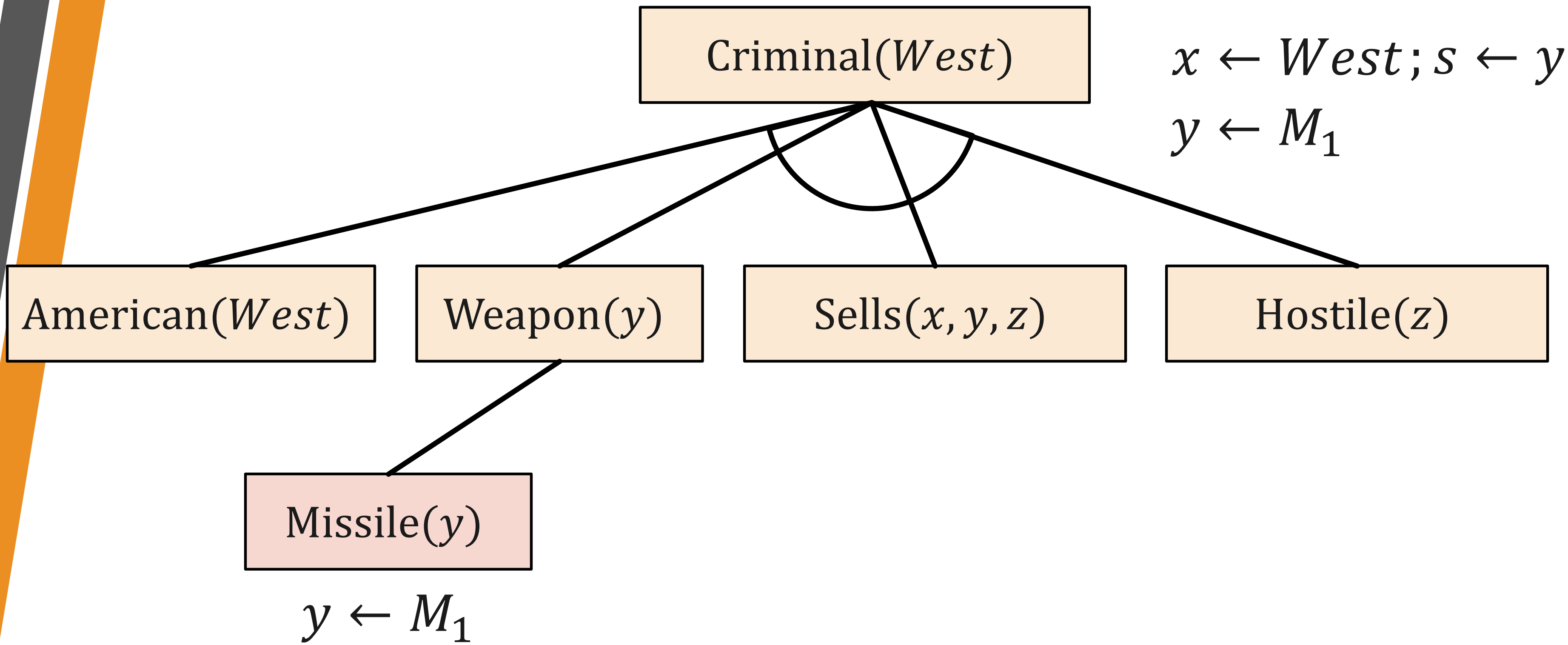
Backward Chaining: Proof Tree

$x \leftarrow West\,; s \leftarrow y$



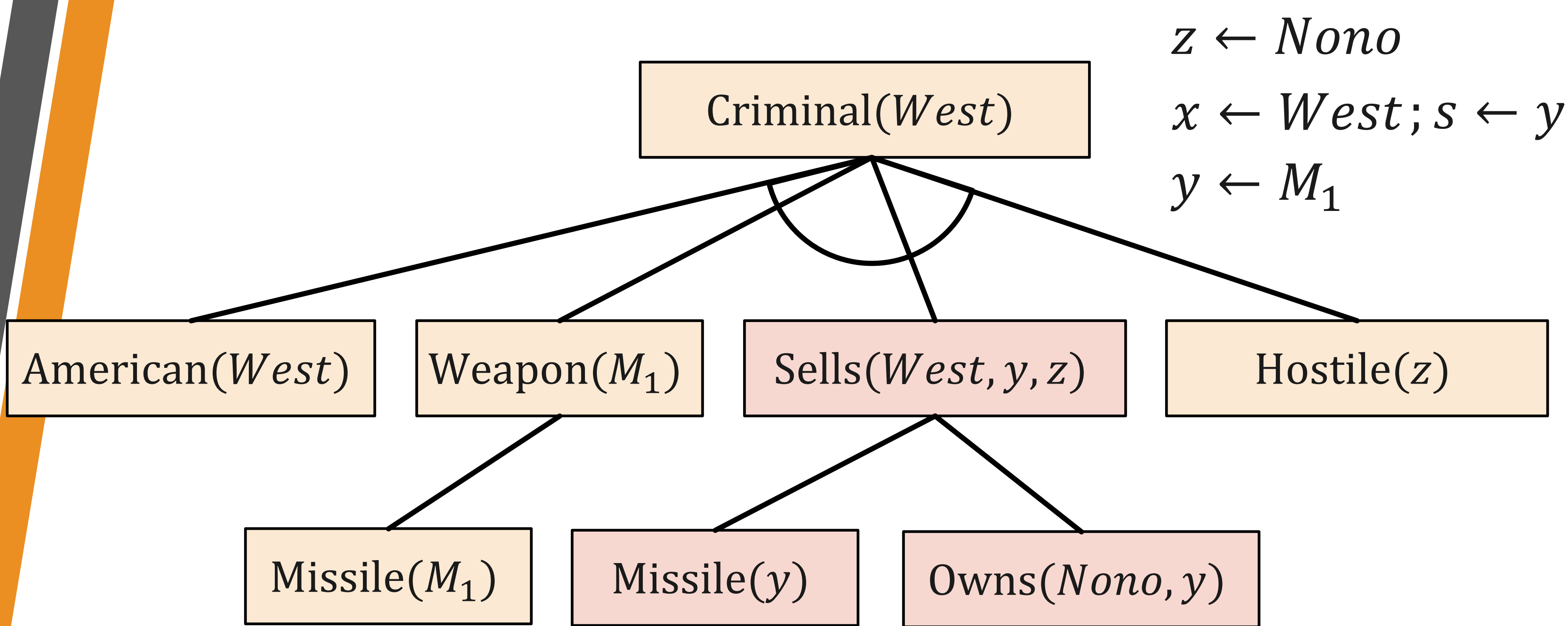
Unifying we get $[s \leftarrow y]$

Backward Chaining: Proof Tree



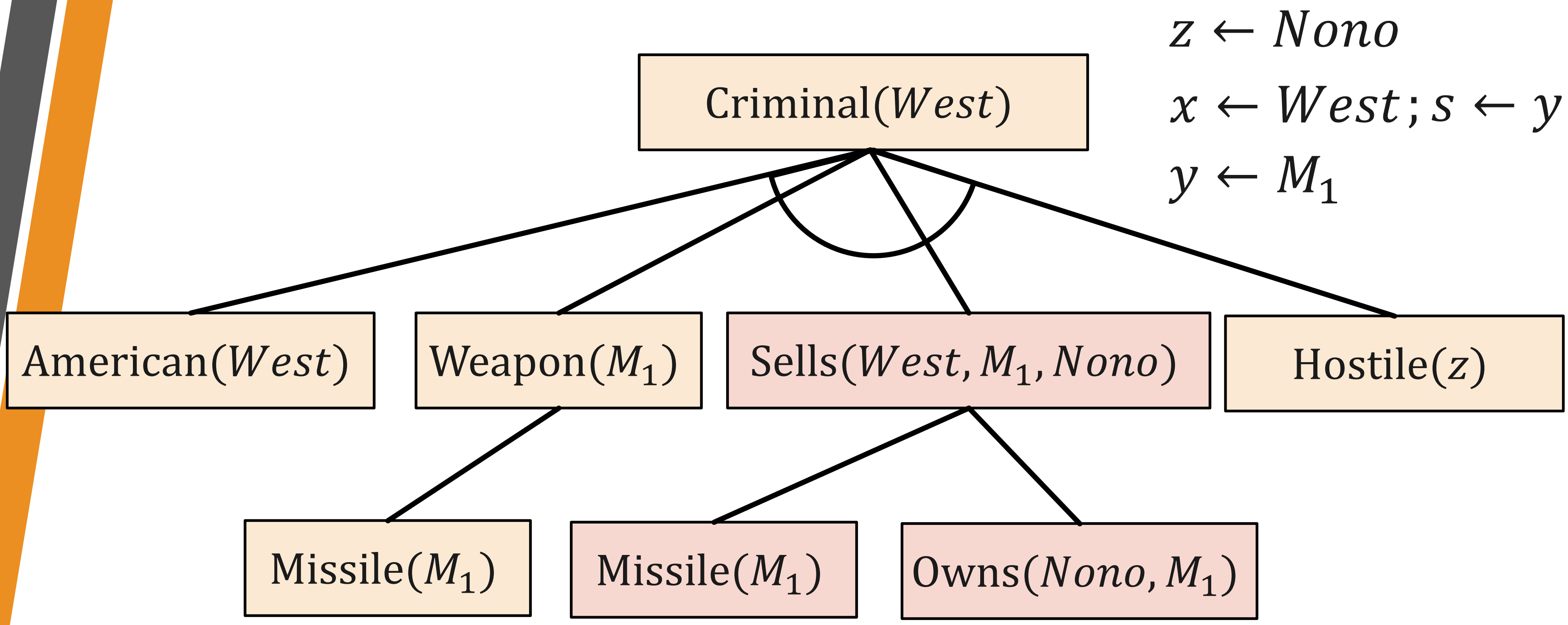
Unify on Missile(M_1) and Missile(y) outputs $y \leftarrow M_1$.

Backward Chaining: Proof Tree



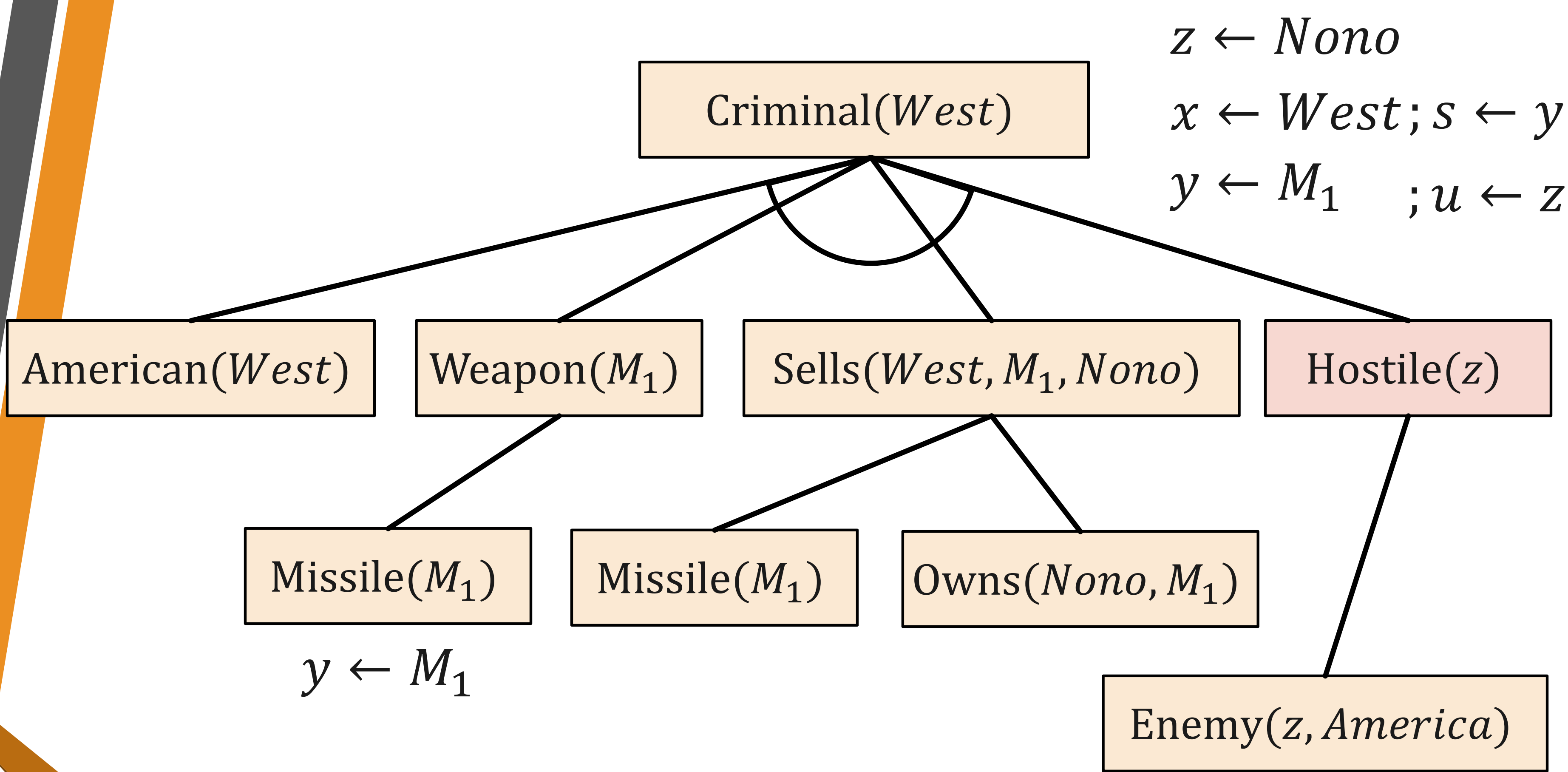
Fetch $\text{Missile}(t) \wedge \text{Owns}(Nono, t) \Rightarrow \text{Sells}(West, t, Nono)$
UNIFY to get $t \leftarrow y$ and $z \leftarrow Nono$

Backward Chaining: Proof Tree



no contradiction with $y \leftarrow M_1$ so we are good!

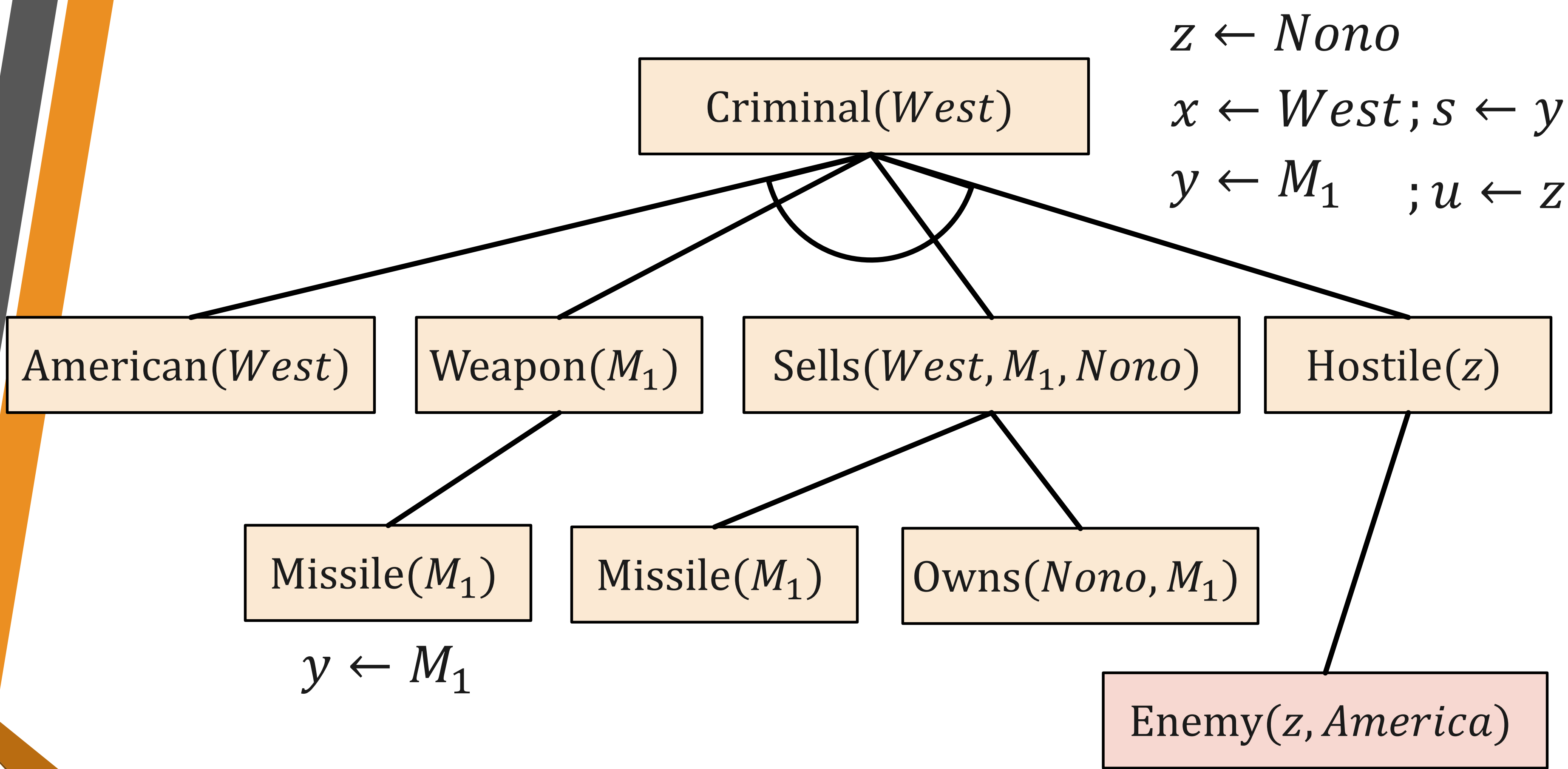
Backward Chaining: Proof Tree



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

UNIFY to get $u \leftarrow z$

Backward Chaining: Proof Tree



UNIFY with $\text{Enemy}(Nono, America)$, get no contradiction with current assignment to z.

Properties of Backward Chaining

- Depth-first 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 by caching solutions to previous subgoals
- Widely used for logic programming

Resolution in FOL: Convert to CNF

"Everyone who loves all animals is loved by someone"

$$\forall x: \big(\forall y: \mathrm{Animal}(y) \Rightarrow \mathrm{Loves}(x, y)\big) \Rightarrow \exists y: \mathrm{Loves}(y, x)$$

- **Eliminate implications: $\boldsymbol{A \Rightarrow B}$ becomes $\boldsymbol{\neg A \lor B}$**

$$\forall x: \neg\big(\forall y: \neg\mathrm{Animal}(y) \lor \mathrm{Loves}(x, y)\big)$$
$$\lor\ \exists y: \mathrm{Loves}(y, x)$$

- **De Morgan's rule:**
  $\boldsymbol{\neg\forall x: P \equiv \exists x: \neg P}$ and $\boldsymbol{\neg\exists x: P \equiv \forall x: \neg P}$

$$\forall x: \Big(\exists y: \neg\big(\neg\mathrm{Animal}(y) \lor \mathrm{Loves}(x, y)\big)\Big) \lor \exists y: \mathrm{Loves}(y, x)$$
$$\forall x: \big(\exists y: \neg\neg\mathrm{Animal}(y) \land \neg\mathrm{Loves}(x, y)\big) \lor \exists y: \mathrm{Loves}(y, x)$$
$$\forall x: \big(\exists y: \mathrm{Animal}(y) \land \neg\mathrm{Loves}(x, y)\big) \lor \exists y: \mathrm{Loves}(y, x)$$

Resolution in FOL: Convert to CNF

"Everyone who loves all animals is loved by someone"

$$\forall x: \big(\forall y: \text{Animal}(y) \Rightarrow \text{Loves}(x, y)\big) \Rightarrow \exists y: \text{Loves}(y, x)$$

- **Standardize variables:**

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

- **Skolemize existential quantifiers:** replace with functions **depending on external universal quantifier**. Cannot just apply existential instantiation (y and z may depend on x)!

  $$\forall x: \Big(\text{Animal}\big(F(x)\big) \wedge \neg\text{Loves}\big(x, F(x)\big)\Big) \vee \text{Loves}\big(G(x), x\big)$$

$F(x)$ and $G(x)$ are called Skolem Functions.

Resolution in FOL: Convert to CNF

"Everyone who loves all animals is loved by someone"

$$\forall x: \big(\forall y: \text{Animal}(y) \Rightarrow \text{Loves}(x, y)\big) \Rightarrow \exists z: \text{Loves}(z, x)$$

- **Drop Universal Quantifiers:**

$$\Big(\text{Animal}\big(F(x)\big) \land \neg\text{Loves}\big(x, F(x)\big)\Big) \lor \text{Loves}(G(x), x)$$

- **Distribute ∨ over ∧:**

$$(A \lor B) \land C \equiv A \land C \lor A \land B$$

$$\Big(\text{Animal}\big(F(x)\big) \lor \text{Loves}(G(x), x)\Big) \land \Big(\neg\text{Loves}\big(x, F(x)\big) \lor \text{Loves}(G(x), x)\Big)$$

CNF Resolution in FOL

- FOL literals are **complements** if one unifies with negation of the other. $F(x,y), \neg F\big(u, G(z)\big)$ are complements with $\theta = [x \leftarrow u, y \leftarrow G(z)]$

- Suppose that a, b are complements that can be unified with θ, then the two clauses $\left(\ell_1 \vee \cdots \vee \ell_q \vee a\right)$ and $(m_1 \vee \cdots \vee m_r \vee b)$ resolve to

$$SUBST\left(\theta, \ell_1 \vee \cdots \vee \ell_q \vee m_1 \vee \cdots \vee m_r\right)$$

  The clauses are **standardized apart**: share no variables

- For example,

$$\frac{\neg\mathrm{Loves}(u,v) \vee \neg\mathrm{Kills}(u,v); \mathrm{Animal}\big(F(x)\big) \vee \mathrm{Loves}(G(x),x)}{\neg Kills(G(x),x) \vee \mathrm{Animal}\big(F(x)\big)}$$

  where $\mathrm{UNIFY}\big(\neg\mathrm{Loves}(u,v), \neg\mathrm{Loves}(G(x),x)\big) = [u \leftarrow G(x), v \leftarrow x]$

CNF Resolution in FOL

- First-order factoring: removes redundant literals by reducing 2 literals to one if they are unifiable.
- For example,

  $\big(P(x) \vee G(a,b)\big)$;$\big(\neg P(y) \vee G(k,\ell)\big)$ results in $G(a,b) \vee G(k,\ell)$.

  - Apply factoring to get $G(k,\ell)$
- To prove $KB \vDash \alpha$, show that $KB \wedge \neg\alpha$ results in a contradiction.

Example KB in FOL

1. $\neg\text{American}(x) \vee \neg\text{Weapon}(y) \vee \neg\text{Hostile}(z) \vee \neg\text{Sells}(x, y, z) \vee \text{Criminal}(x)$
2. $\text{Enemy}(Nono, America)$
3. $\text{Owns}(Nono, M_1) \wedge \text{Missile}(M_1)$
4. $\neg\text{Missile}(q) \vee \neg\text{Owns}(Nono, q) \vee \text{Sells}(West, q, Nono)$
5. $\text{American}(West)$
6. $\neg\text{Missile}(r) \vee \text{Weapon}(r)$
7. $\neg\text{Enemy}(s, America) \vee \text{Hostile}(s)$

Query: $\alpha = \text{Criminal}(West)$

Add: $\neg\alpha = \neg\text{Criminal}(West)$ to KB, find contradiction

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

$x \leftarrow West$

$American(West)$ | $\neg American(West) \lor \neg Weapon(y) \lor \neg Hostile(z) \lor \neg Sells(West, y, z)$

$\neg Missile(r) \lor Weapon(r)$ | $\neg Weapon(y) \lor \neg Hostile(z) \lor \neg Sells(West, y, z)$

$r \leftarrow y$

$Missile(M_1)$ | $\neg Missile(y) \lor \neg Hostile(z) \lor \neg Sells(West, y, z)$

$y \leftarrow M_1$

$\neg Missile(q) \lor \neg Owns(Nono, q) \lor Sells(West, q, Nono)$ | $\neg Hostile(z) \lor \neg Sells(West, M_1, z)$

$q \leftarrow M_1$

$z \leftarrow Nono$

$Missile(M_1)$ | $\neg Hostile(Nono) \lor \neg Missile(M_1) \lor \neg Owns(Nono, M_1)$

$Owns(Nono, M_1)$ | $\neg Hostile(Nono) \lor \neg Owns(Nono, M_1)$

$\neg Enemy(s, America) \lor Hostile(s)$ | $\neg Hostile(Nono)$

$s \leftarrow Nono$

$Enemy(Nono, America)$ | $\neg Enemy(Nono, America)$