Lecture 9: Inference in First Order Logic

Artificial Intelligence

CS-6364

Outline

- Universal/Existential Instantiation
- *Reducing FOL inference to propositional inference*
- Unification & Lifting
- Generalized Modus Ponens
- Forward chaining
- Backward chaining
- Resolution

Inference in FOL

- Inference rules for quantifiers
  - Universal Instantiation
  - Existential Instantiation
- Reduction to propositional inference
- Unification and lifting

A brief history of reasoning

450B.C.	Stoics	propositional logic, inference (maybe)
322B.C.	Aristotle	"syllogisms" (inference rules), quantifiers
1565	Cardano	probability theory (propositional logic + uncertainty)
1847	Boole	propositional logic (again)
1879	Frege	first-order logic
1922	Wittgenstein	proof by truth tables
1930	Gödel	$\exists$ complete algorithm for FOL
1930	Herbrand	complete algorithm for FOL (reduce to propositional)
1931	Gödel	$\neg\exists$ complete algorithm for arithmetic
1960	Davis/Putnam	"practical" algorithm for propositional logic
1965	Robinson	"practical" algorithm for FOL—resolution

Universal Instantiation

- This rule says that we can infer any sentence obtained by substituting a ground term (a term without variables) for a variable
- **Substitution or binding list** is a set of variable/term pairs (remember Lecture 8 slide 35!!!)
- SUBST(θ, α) denotes the result of applying the substitution θ to sentence α
  - Example: α: $\forall x,y\ \ D(x) \Rightarrow Q(x,y) \wedge R(y)$

    $\theta = x/A$

    $SUBST(\theta, \alpha) = \forall y\ P(A) \Rightarrow Q(A,y) \wedge R(y)$
- **Universal Instantiation**:

for any variable v and ground term g

$$\frac{\forall v \alpha}{SUBST(\{v/g\},\alpha)}$$

$\forall$x Young(x)$\wedge$Beautiful(x)$\Rightarrow$Attractive(x)

SUBST(x/Robert) :
Young(Robert) $\wedge$Beautiful(Robert)$\Rightarrow$Attractive(Robert)

Universal instantiation (UI)

- *Every instantiation of a universally quantified sentence is entailed by it:* for any variable *v* and ground term *g*

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

Example: $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ yields:

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

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

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

Existential Instantiation

For any sentence alpha, variable *v*, and constant symbol *K* that <u>does not appear anywhere else in the knowledge base</u>

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

<u>Example</u>: $\exists$x Crown(x)$\wedge$OnHead(x,John)

we can infer:

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

as long as C_1 doesn't appear anywhere else in the KB

Existential instantiation (EI)

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

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

Example $\exists x\ Crown(x) \wedge OnHead(x, John)$ yields:

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

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

Existential & Universal Instantiation

- Universal Instantiation (UI) can be applied several times to **add** new sentences to the KB
  - The new KB' is logically equivalent to KB (the old one)

- Existential Instantiation (EI) can be applied once to **replace** an existential sentence
  - The new KB' is not logically equivalent to the old one, but it is satisfiable iff the old KB was satisfiable. However, it is **inferentially equivalent**!!!

Logical equivalence: two sentences α and β are logically equivalent *if they are true in the same set of models.* $\alpha \equiv \beta$ if $\alpha \models \beta$ and $\beta \models \alpha$

Validity: a sentence is valid if it is true in all models. Valid sentences are also known as *tautologies.*

Satisfiability: a sentence is satisfiable if it is true in some model.

Reduction to propositional inference

Suppose the KB contains:

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

King(John)

Greedy(John)

Brother(Richard,John)

- When applying Universal Instantiation with all possible ground terms (e.g. John, Richard)   with {x/John}, with {x/Richard}

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

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

King(John)

Greedy(John)

Brother(Richard,John)

all are propositional sentences!

The new KB is propositionalized: proposition symbols are

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

Reduction contd.

- Every FOL KB can be propositionalized so as to preserve entailment - *Theorem by Jacques Herbrand (1930)*

- A ground sentence is entailed by new KB iff entailed by original KB
- Problem: with function symbols, there are infinitely many ground terms

Idea: propositionalize KB and query, apply resolution, return result, BUT

- **Theorem:** Turing (1936), Church (1936) Entailment for FOL is semidecidable (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every non-entailed sentence.)

Unification and Lifting

- A FOL inference Rule: Generalized Modus Ponens

Suppose you have the KB:

$\forall x$ *King(x)* $\wedge$ *Greedy(x)* $\Rightarrow$ *Evil(x)*

King(John)

Greedy(John)

Brother(Richard, John)

QUERY:
Evil(x)

θ = *x/John*

{*King(John), Greedy(John),*
AND-INTRODUCTION: King(John) $\wedge$ *Greedy(John),*
King(John) $\wedge$ *Greedy(John)* $\Rightarrow$ *Evil(John) AFTER* θ
MODUS PONENS: Evil(John)}

Generalized Modus Ponens-1

Suppose now you have the KB:

GMP used with KB of definite clauses (exactly one literal)

$\forall x$ *King(x)* ∧ *Greedy(x)* ⇒ *Evil(x)*

King(John)

$\forall y$ *Greedy(y)*

Brother(Richard, John)

QUERY:
Evil(x)

θ = *{x/John, y/John}*

{King(John),
Greedy(John), AFTER θ
AND-INTRODUCTION: King(John) ∧ *Greedy(John),*
King(John) ∧ *Greedy(John)* ⇒ *Evil(John) AFTER* θ
MODUS PONENS: Evil(John)}

For atomic sentence p_i, p_i' and q such that $SUBST(\theta, p_i') = SUBST(\theta, p_i)$ for all i

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

Generalized Modus Ponens-2

For atomic sentence p_i, p_i' and q such that $SUBST(\theta, p_i') = SUBST(\theta, p_i)$ for all i

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

$n+1$ *premises*

Result of applying θ

θ = *{x/John, y/John}*

QUERY:
Evil(x)

p_1' is *King(John)* | p_1 is *King(x)*

p_2' is *Greedy(y)* | p_2 is *Greedy(x)*

$SUBST(\theta, q)$ is *Evil(John)* | q is *Evil(x)*

Universal Instantiation through θ

Universal Instantiation through θ

the KB:

∀x *King(x)* ∧ *Greedy(x)* ⇒ *Evil(x)*

King(John)

∀y *Greedy(y)*

Brother(Richard, John)

$In\ GMP$: $SUBST(\theta, p_i')$= $SUBST(\theta, p_i)$ for every *i*

Generalized Modus Ponens-3

Generalized Modus Ponens (GMP) is a **lifted version** of Modus Ponens from Propositional logic to FOL.

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

$$\frac{\alpha \Rightarrow \beta\ ,\ \alpha}{\beta}$$

The advantage of lifted inference rules over propositionalization is that they make only those substitutions that are required for particular inferences to proceed!

*Lifted inference require **finding substituions** that make different logical expressions look identical!!! This process is called **unification**.*

Unification-1

- We could get the answer to a query through inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

$\theta = \{x/John, y/John\}$ worked!!!!

We can unify two sentences α and β, and write $Unify(\alpha,\beta) = \theta$

if $SUBST(\theta, \alpha) = SUBST(\theta,\beta)$

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-2

- How do we find the substitution through Unification???

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-3

- How do we find the substitution through Unification???

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-4

- How do we find the substitution through Unification???

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-5

- How do we find the substitution through Unification???

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}

- SOLUTION: Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification - MGU

- To unify *Knows(John,x)* and *Knows(y,z)*,
  θ = {y/John, x/z } or θ = {y/John, x/John, z/John}

- The first unifier is more general than the second.
- There is a single most general unifier (MGU) that is unique up to renaming of variables.
  MGU = { y/John, x/z }

Unification - again

- Inference rules require finding substitutions that make different logical expressions look identical
  - The process is called unification and it is a key component of all first-order inference algorithms

UNIFY(p,q)=θ where SUBST(θ,p)=SUBST(θ,q)

↳ *The UNIFY algorithm takes two sentences and returns a unifier for one of them*

<u>Examples:</u>

UNIFY(knows(John,x),knows(John,Jane))={x/Jane}

UNIFY(knows(John,x),knows(y,Bill))={x/Bill,y/John}

UNIFY(knows(John,x),knows(y,Mother(y)))
={y/John,x/Mother(John)}

UNIFY(knows(John,x),knows(x,Elizabeth))=fail

Unification Example

- *Suppose we have a rule*

  $\boxed{\text{Knows(John,x)} \Rightarrow \text{Hates(John,x)}}$

  "John hates everyone he knows"

?? We want to use this rule with the Modus Ponens inference rule to find whom he hates (x=?)

how?

We need to find those sentences in the KB that Unify with Knows(John,x) and then apply the unifier to Hates(John,x)

Working the MGU

- Let the KB contain:
  - Knows(John, Jane)
  - Knows(y, Leonid)
  - Knows(y, Mother(y))
  - Knows(x, Elisabeth)

- *Unifying the antecedent of the rule Knows(John,x) ⇒ Hates(John,x) against each of the sentences in the KB:*

UNIFY(Knows(John,x),Knows(John,Jane))={x/Jane}

UNIFY(Knows(John,x),Knows(y,Leonid))=
{x/Leonid, y/John}

UNIFY(Knows(John,x),Knows(y,Mother(y)))=
{y/John, x/Mother(John)}

UNIFY(Knows(John,x),Knows(x,Elisabeth))=fail

Standardizing Apart

Renaming variables to avoid name clashes

For example, rename x in

knows(x,Elizabeth) to z_{17}

→ UNIFY(knows(John,x),knows(z_{17},Elizabeth))= {x/Elizabeth,z_{17}/John}

<u>Further examples:</u>

UNIFY(knows(John,x),knows(y,z))

can return θ_1 = {y/John,x/John,z/John}

θ_2 = {y/John,x/z,x/John}

θ_1 yields knows(John,z)

θ_2 yields knows(John,John)

→ θ_1 is more general than θ_2

For every pair of FOL atomic sentences there is a single Most General Modifier (MGU)

The most general unifier (MGU)

- MGU is the substitution that makes the least commitment about the binding of the variables
- For example

UNIFY(Knows(John,x), Knows(y,z)) = {y/John, x/z}

or {y/John, x/z, w/Treda}

or {y/John, x/John, z/John}

or ...

MGU = {y/John, x/z}

The MGU iterative Algorithm

function UNIFY(x,y,θ) **returns** a ***substitution*** to make x and y identical
  inputs: *x*, a variable, constant, list or compound
    y, a variable, constant, list or compound
    θ, the substitution built up so far (optional, defaults to empty)

if *θ* = failure **then return** *false*
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(ARGS[*x*], ARGS[*y*], UNIFY(OP[*x*], OP[*y*], *θ*))
else if LIST?(*x*) **and** LIST?(*y*) **then**
    return UNIFY(REST[*x*], REST[*y*], UNIFY(FIRST[*x*],FIRST[*y*],*θ*)
else return *failure*

---

function UNIFY-VAR(var,x,θ) **returns** a ***substitution***
inputs: *var*, a variable
  x, any expression
  θ, the substitution built up so far

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

Forward Chaining

<u>The idea:</u> Start with the atomic sentences in the KB and apply Modus Ponens in the forward direction, adding new atomic sentences until no further inferences can be Made.

Situation: 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.

- What do we want to prove?

  $\Rightarrow$ West is a criminal

FOL representation

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

P.1 $American(x) \wedge Weapon(y) \wedge Nation(z) \wedge Hostile(z) \wedge Sells(x,y,z) \Rightarrow Criminal(x)$

"... Country Nono...has some

P.2 $Owns(Nono,M_1)$

$Missile(M_1)$

$Nation(Nono)$

"All of its missiles were sold by Colonel West":

P.3 $Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$

Common sense knowledge

We need also to know that missiles are weapons:

P.4 $\forall x\ Missile(x) \Rightarrow Weapons(x)$

An enemy of America counts as "hostile"

P.5 $\forall x\ Enemy(x,America) \Rightarrow Hostile(x)$

"...West who is American..."

P.6 American(West)

"... Nono, an enemy of America..."

P.7 Enemy(Nono, America)

A 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 sentence r in KB do
        (p1 ∧ … ∧ pn ⇒ q) ← STANDARDIZE-APART(r)
        for each θ such that SUBST(θ,p1∧ … ∧ pn)= SUBST(θ,p1′∧ … ∧ pn′)
                                          for some p1′, …, pn′ in KB
            q′ ← SUBST(θ,q)
            if q′ is not a renaming of some 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
```

How is it working on an example?

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 sentence r in KB do
        (p1 ∧ … ∧ pn ⇒ q) ← STANDARDIZE-APART(r)
        for each θ such that SUBST(θ,p1∧ … ∧ pn)= SUBST(θ,p1′∧ … ∧ pn′)
                                   for some p1′, …, pn′ in KB
            q′← SUBST(θ,q)
            if q′ is not a renaming of some 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
```

α

"Is West a criminal ????"

Criminal(West)

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

P.1 American(x) $\wedge$ Weapon(y) $\wedge$ Nation(z) $\wedge$ Hostile(z) $\wedge$ Sells(x,y,z) $\Rightarrow$ Criminal(x)

"... Country Nono...has some missiles":

P.2 Owns(Nono,M_1)
Missile (M_1)
Nation(Nono)

"All of its missiles were sold by Colonel West":

P.3 Missile(x) $\wedge$ Owns(Nono,x) $\Rightarrow$ Sells(West,x,Nono)

We need also to know that missiles are weapons:

P.4 $\forall$x Missile(x) $\Rightarrow$ Weapons(x)

An enemy of America counts as "hostile"

P.5 $\forall$x Enemy(x,America) $\Rightarrow$ Hostile(x)

"...West who is American..."

P.6 American(West)

"... Nono, an enemy of America..."

P.7 Enemy(Nono, America)

KB

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.

Starting Point????

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 sentence *r* **in** *KB* **do**
($p_1 \wedge \ldots \wedge p_n \Rightarrow q$) $\leftarrow$ **STANDARDIZE-APART**(r)
for each θ such that **SUBST**(θ,$p_1 \wedge \ldots \wedge p_n$)= **SUBST**($\theta$,$p_1' \wedge \ldots \wedge p_n'$)
for some p_1', ..., p_n' **in** *KB*
q' $\leftarrow$ **SUBST**(θ,*q*)
if *q'* is not a renaming of some sentence already in *KB* or *new* **then do**
add *q'* to *new*
$\varphi \leftarrow$ **UNIFY**(*q'*, α)
if φ is not fail **then return** φ
add *new* to *KB*
return *false*

All Implication sentences from the KB

Find substitutions from the KB for the preconditions of the implication sentences

All Implication sentences from the KB

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

P.1 American(x) $\wedge$ Weapon(y) $\wedge$ Nation(z) $\wedge$ Hostile(z) $\wedge$ Sells(x,y,z) $\Rightarrow$ Criminal(x)

"All of its missiles were sold by Colonel West":

P.3 Missile(y) $\wedge$ Owns(Nono,y) $\Rightarrow$ Sells(West,y,Nono)

We need also to know that missiles are weapons:

P.4 $\forall$x Missile(y) $\Rightarrow$ Weapons(y)

An enemy of America counts as "hostile"

P.5 $\forall$x Enemy(z,America) $\Rightarrow$ Hostile(z)

P.1 American(x) $\wedge$ Weapon(y) $\wedge$ Nation(z) $\wedge$ Hostile(z) $\wedge$ Sells(x,y,z) $\Rightarrow$ Criminal(x)
P.6 American(West)
θ1: x/West

P.3 Missile (M_1)$\wedge$ Owns(Nono,M_1) $\Rightarrow$ Sells(West,M_1,Nono)
θ2: y/M1
P.2 Missile(M_1)
P.4 Missile(M_1) $\Rightarrow$ Weapons(M_1)

P.5 $\forall$x Enemy(z,America) $\Rightarrow$ Hostile(z)
P.7 Enemy(Nono, America)
θ3: z/Nono

Next????

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 sentence *r* **in** *KB* **do**
($p_1 \wedge \ldots \wedge p_n \Rightarrow q$) $\leftarrow$ **STANDARDIZE-APART**(r)
for each θ such that **SUBST**(θ,$p_1 \wedge \ldots \wedge p_n$)= **SUBST**($\theta$,$p_1' \wedge \ldots \wedge p_n'$)
for some p_1', …, p_n' **in** *KB*
$q' \leftarrow$ **SUBST**(θ,q)
if q' is not a renaming of some sentence already in *KB* or *new* **then do**
add q' to *new*
$\varphi \leftarrow$ **UNIFY**(q', α)
if φ is not fail **then return** φ
add *new* to *KB*
return *false*

Generate **new conclusions** from All Implication sentences from the KB, using the corresponding substitutions

All Implication sentences from the KB

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

P.1 American(x) $\wedge$ Weapon(y) $\wedge$ Nation(z) $\wedge$ Hostile(z) $\wedge$ Sells(x,y,z) $\Rightarrow$ Criminal(x)

"All of its missiles were sold by Colonel West":

P.3 Missile(y) $\wedge$ Owns(Nono,y) $\Rightarrow$ Sells(West,y,Nono)

We need also to know that missiles are weapons:

P.4 $\forall$x Missile(y) $\Rightarrow$ Weapons(y)

An enemy of America counts as "hostile"

P.5 $\forall$x Enemy(z,America) $\Rightarrow$ Hostile(z)

New (1) – first iteration

P.1 American(West) $\wedge$ Weapon(y) $\wedge$ Nation(z) $\wedge$ Hostile(z) $\wedge$ Sells(West,y,z) $\Rightarrow$ Criminal(West)

θ1: x/West

Has unsatisfied premises!

P.3 Missile(M1) $\wedge$ Owns(Nono,M1) $\Rightarrow$ Sells(West,M1,Nono)

θ2: y/M1

P.4 $\forall$x Missile(M1) $\Rightarrow$ Weapons(M1) → q': Weapons(M1)

P.5 Enemy(Nono,America) $\Rightarrow$ Hostile(Nono) → q': Hostile(Nono)

θ3: z/Nono

Next????

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 sentence *r* **in** *KB* **do**
$(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ **STANDARDIZE-APART**(r)
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' is not a renaming of some sentence already in *KB* or *new* **then do**
add q' to *new*
$\varphi \leftarrow$ **UNIFY**(q', α)
if φ is not fail **then return** φ
add *new* to *KB*
return *false*

Generate **new conclusions** from All Implication sentences from the KB, using the corresponding substitutions

All Implication sentences from the KB

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

P.1 American(x) $\wedge$ Weapon(y) $\wedge$ Nation(z) $\wedge$ Hostile(z) $\wedge$ Sells(x,y,z) $\Rightarrow$ Criminal(x)

"All of its missiles were sold by Colonel West":

P.3 Missile(y) $\wedge$ Owns(Nono,y) $\Rightarrow$ Sells(West,y,Nono)

We need also to know that missiles are weapons:

P.4 $\forall$x Missile(y) $\Rightarrow$ Weapons(y)

An enemy of America counts as "hostile"

P.5 $\forall$x Enemy(z,America) $\Rightarrow$ Hostile(z)

New (2) – second iteration

P.1 American(West) $\wedge$ Weapon(y) $\wedge$ Nation(z) $\wedge$ Hostile(z) $\wedge$ Sells(West,y,z) $\Rightarrow$ Criminal(West)

θ1: x/West

Has unsatisfied premises!

P.2

P.3 Missile(M1) $\wedge$ Owns(Nono,M1) $\Rightarrow$ Sells(West,M1,Nono) → q': Sells(West,M1,Nono))

θ2: y/M1

P.4 $\forall$x Missile(M1) $\Rightarrow$ Weapons(M1) → q': Weapons(M1)

P.5 Enemy(Nono,America) $\Rightarrow$ Hostile(Nono) → q': Hostile(Nono)

θ3: z/Nono

New conclusions

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 sentence *r* **in** *KB* **do**
($p_1 \wedge \ldots \wedge p_n \Rightarrow q$) $\leftarrow$ **STANDARDIZE-APART**(r)
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' is not a renaming of some sentence already in *KB* or *new* **then do**
add q' to *new*
$\varphi \leftarrow$ **UNIFY**(q', α)
if φ is not fail **then return** φ
add *new* to *KB*
return *false*

Generate **new conclusions** from All Implication sentences from the KB, using the corresponding substitutions

All Implication sentences from the KB

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

P.1 American(x) $\wedge$ Weapon(y) $\wedge$ Nation(z) $\wedge$ Hostile(z) $\wedge$ Sells(x,y,z) $\Rightarrow$ Criminal(x)

"All of its missiles were sold by Colonel West":

P.3 Missile(y) $\wedge$ Owns(Nono,y) $\Rightarrow$ Sells(West,y,Nono)

We need also to know that missiles are weapons:

P.4 $\forall$x Missile(y) $\Rightarrow$ Weapons(y)

An enemy of America counts as "hostile"

P.5 $\forall$x Enemy(z,America) $\Rightarrow$ Hostile(z)

New (3) – Beginning Third iteration

θ1: x/West *θ2: y/M1* *θ3: z/Nono*

P.1 American(West) $\wedge$ Weapon(y) $\wedge$ Nation(z) $\wedge$ Hostile(z) $\wedge$ Sells(West,y,z) $\Rightarrow$ Criminal(West)

$\Downarrow$

American(West) $\wedge$ Weapon(M1) $\wedge$ Nation(Nono) $\wedge$ Hostile(Nono) $\wedge$ Sells(West,M1,Nono) $\Rightarrow$ Criminal(West)

All premises satisfied!

q': Sells(West,M1,Nono))

q': Weapons(M1)

q': Hostile(Nono)

q': Criminal(West)

Finally !!!!

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 sentence *r* **in** *KB* **do**
($p_1 \wedge \ldots \wedge p_n \Rightarrow q$) $\leftarrow$ **STANDARDIZE-APART**(r)
for each θ such that **SUBST**(θ,$p_1 \wedge \ldots \wedge p_n$)= **SUBST**($\theta$,$p_1' \wedge \ldots \wedge p_n'$)
for some p_1', …, p_n' **in** *KB*
$q' \leftarrow$ **SUBST**(θ,q)
if q' is not a renaming of some sentence already in *KB* or *new* **then**
do
add q' to *new*
$\varphi \leftarrow$ **UNIFY**(q', α)
if φ is not fail **then return** φ
add *new* to *KB*
return *false*

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

P.1 American(x) $\wedge$ Weapon(y) $\wedge$ Nation(z) $\wedge$ Hostile(z) $\wedge$ Sells(x,y,z) $\Rightarrow$ Criminal(x)

"All of its missiles were sold by Colonel West":

P.3 Missile(y) $\wedge$ Owns(Nono,y) $\Rightarrow$ Sells(West,y,Nono)

We need also to know that missiles are weapons:

P.4 $\forall$x Missile(y) $\Rightarrow$ Weapons(y)

An enemy of America counts as "hostile"

P.5 $\forall$x Enemy(z,America) $\Rightarrow$ Hostile(z)

Unify the new conclusions with the query

α "Is West a criminal ????"
Criminal(West)

φ

New

Criminal(West) the q' from P1 — *θ1: x/West*

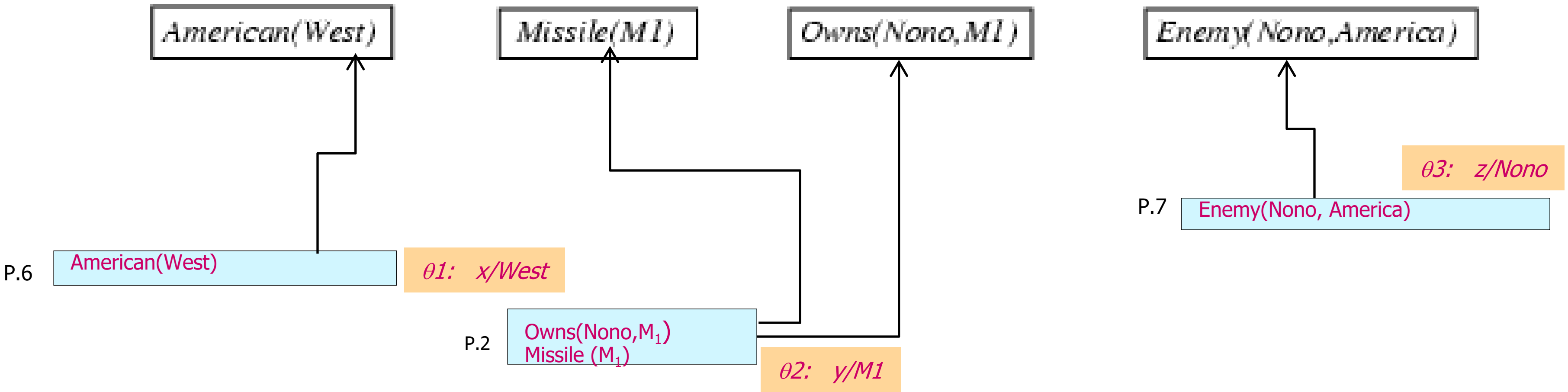
Sells(West, M1, Nono) the q' from P.3

Weapons(M1) the q' from P.4 — *θ2: y/M1*

Hostile(Nono) the q' from P.5 — *θ3: z/Nono*

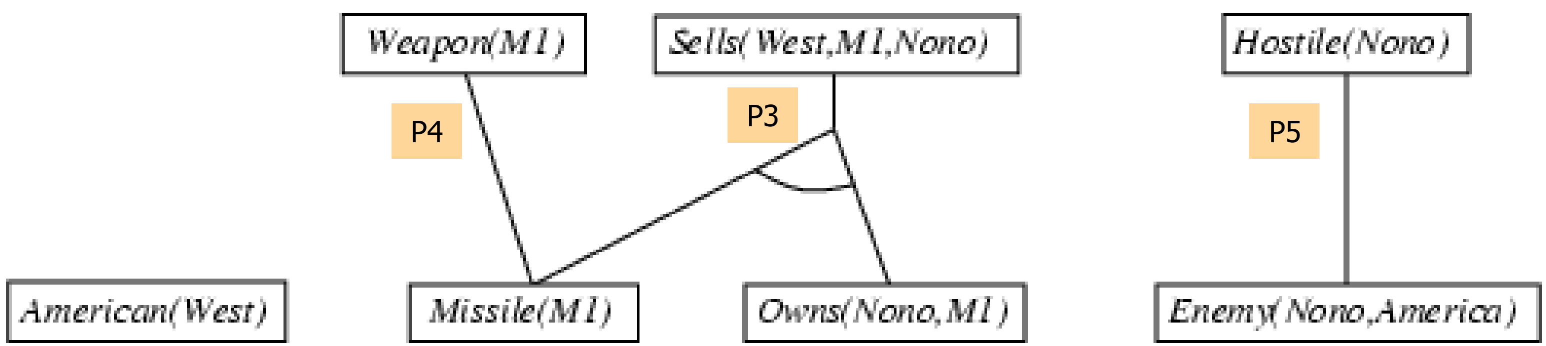
Forward chaining proof

First the substitutions – and the sentences from the KB that granted them



Forward chaining proof

Some new conclusions from NEW – and the sentences from the KB that granted them



We need also to know that missiles are weapons:

P.4 $\forall x$ Missile(x) $\Rightarrow$ Weapons(x)

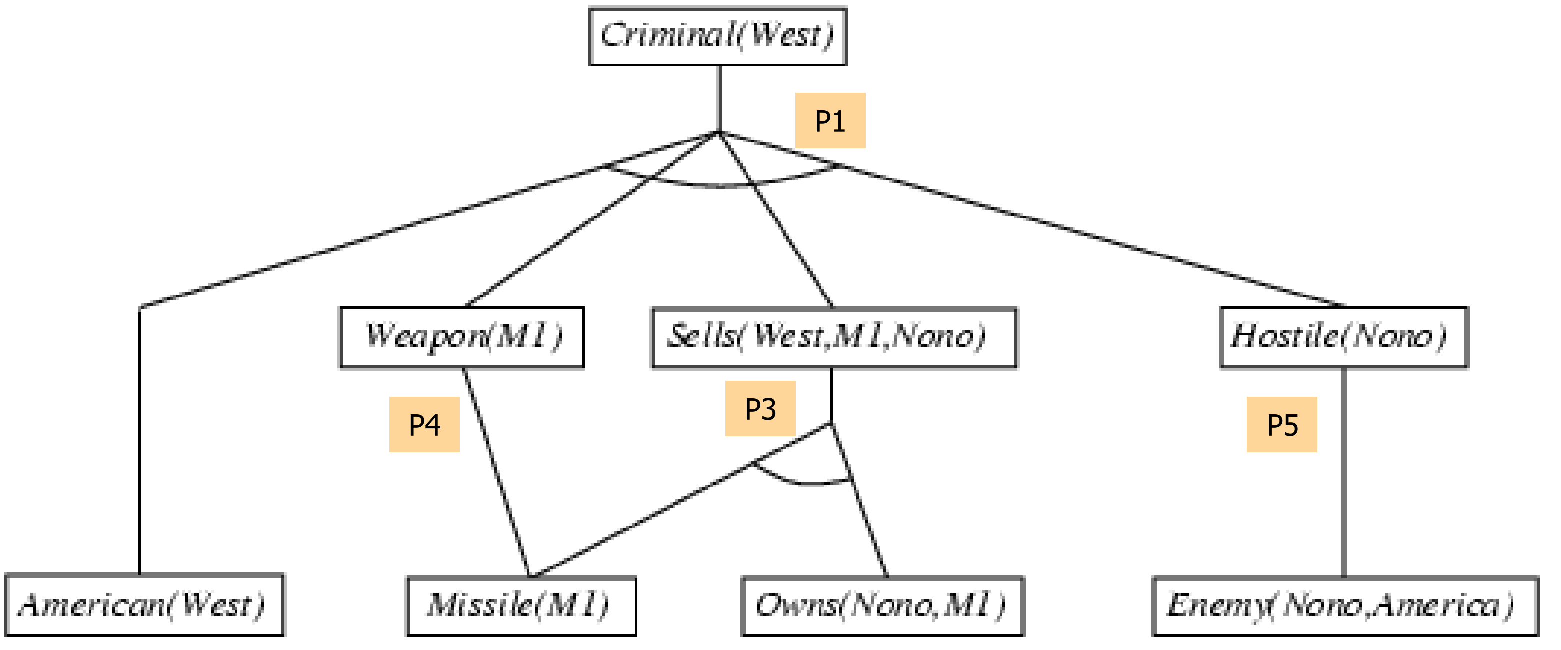
An enemy of America counts as "hostile"

P.5 $\forall x$ Enemy(x,America) $\Rightarrow$ Hostile(x)

"All of its missiles were sold by Colonel West":

P.3 Missile(x) $\wedge$ Owns(Nono,x) $\Rightarrow$ Sells(West,x,Nono)

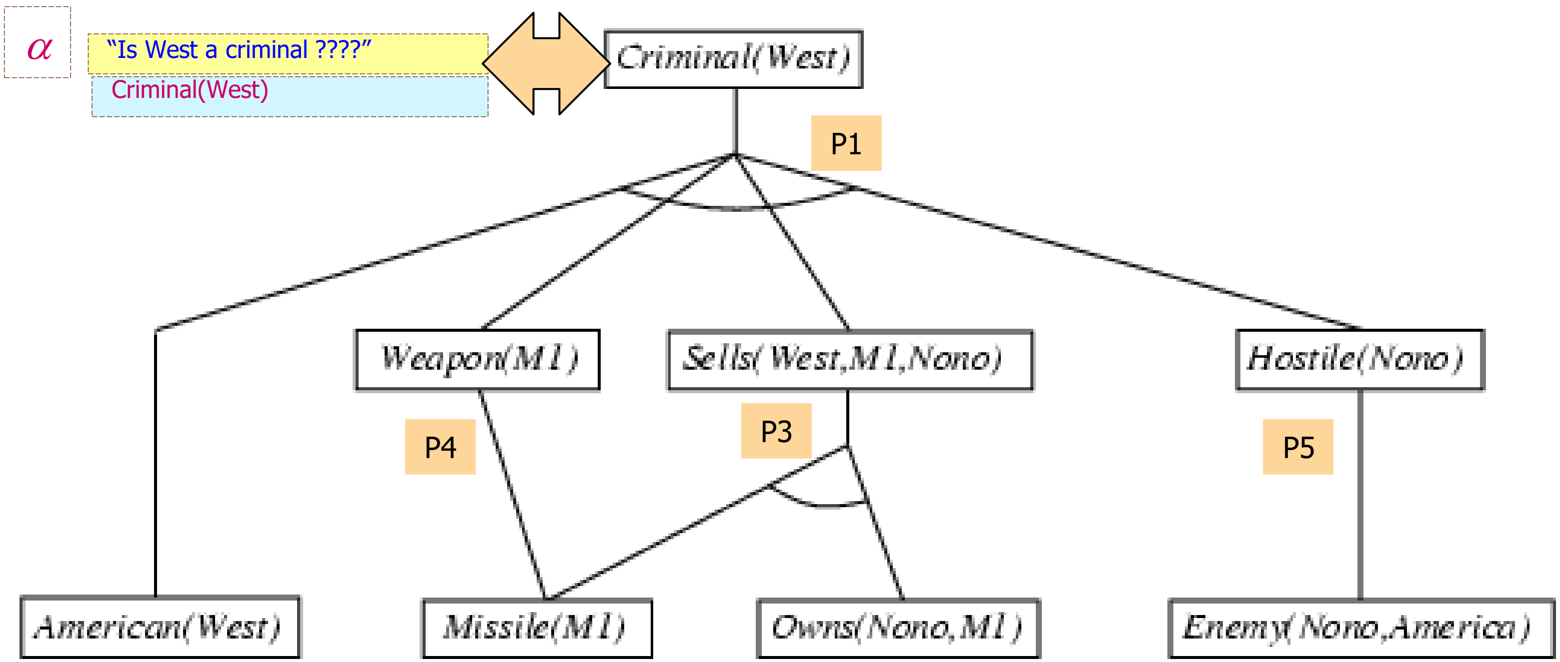
Forward chaining proof



P.1 American(x) ∧ Weapon(y) ∧ Nation(z) ∧ Hostile(z) ∧ Sells(x,y,z) ⇒ Criminal(x)

Forward chaining proof

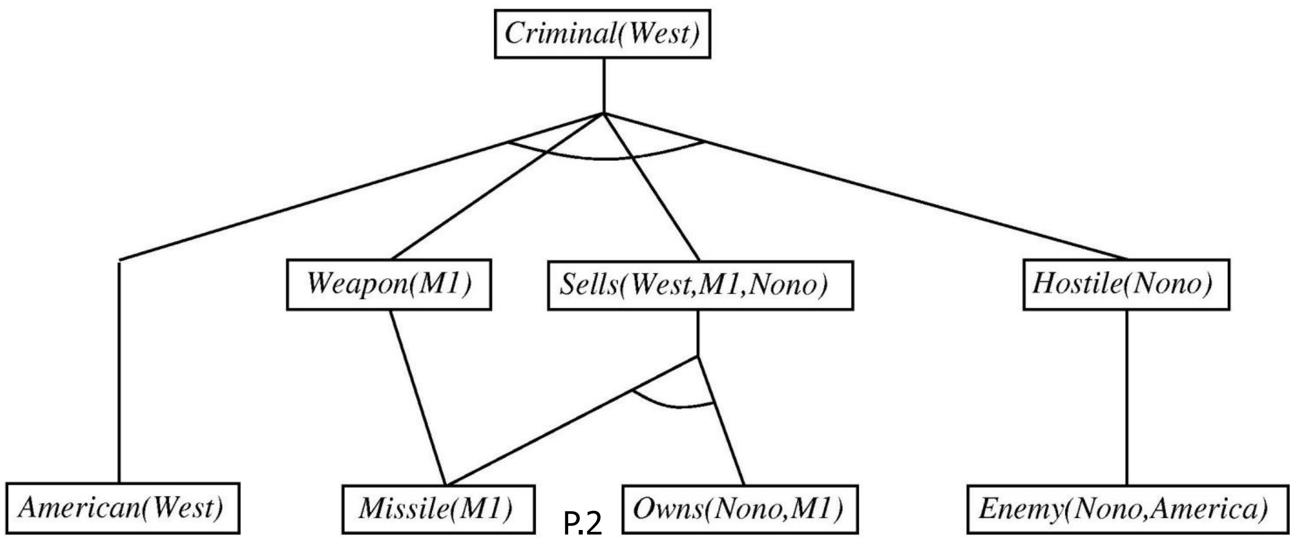
Unification: answer YES



P.1 American(x) $\wedge$ Weapon(y) $\wedge$ Nation(z) $\wedge$ Hostile(z) $\wedge$ Sells(x,y,z) $\Rightarrow$ Criminal(x)

The proof Tree

P.6



P.7

First iteration: the implication sentences

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

P.1 American(x) ∧ Weapon(y) ∧ Nation(z) ∧ Hostile(z) ∧ Sells(x,y,z) ⇒ Criminal(x)

Has unsatisfied Premises in first iteration
On the second iteration It is satisfied with x/West Y/M1, z/Nono

"All of its missiles were sold by Colonel West":

P.3 Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)

We need also to know that missiles are

P.4 ∀x Missile(x) ⇒ Weapons(x)

An enemy of America counts as

P.5 ∀x Enemy(x,America) ⇒ Hostile(x)

Properties of forward chaining

- Sound and complete for first-order definite clauses

Datalog = first-order definite clauses + no functions

FC terminates for Datalog in finite number of iterations

- May not terminate in general if *α* is not entailed

- This is unavoidable: entailment with definite clauses is semidecidable

Efficiency of forward chaining

Incremental forward chaining: 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 positive literal

Matching itself can be expensive:

Database indexing allows O(1) retrieval of known facts

- e.g., query *Missile(x)* retrieves *Missile*(M_1)
-

Forward chaining is widely used in deductive databases

A Backward-chaining Algorithm

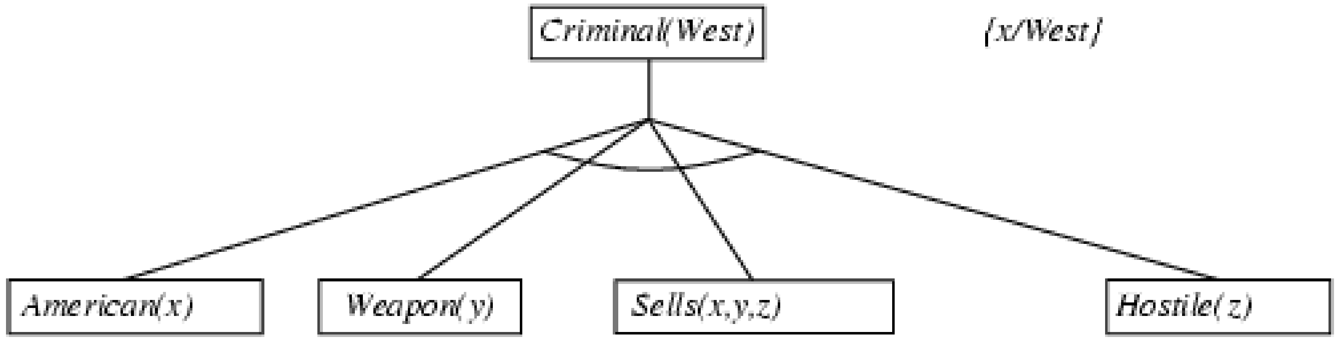
function FOL-BC-ASK(KB,goals,θ) **returns** a ***set of substitutions***
inputs: *KB*, a knowledge base
goals, a list of conjuncts forming a query (θ already applied)
θ, the current substitution, initially the empty substitution
local variables: *answers*, a set of substitutions, initially empty

if *goals* is empty then return {*θ*}
q′ ← SUBST(*θ*,FIRST(*goals*))
for each sentence *r* **in** *KB*
where **STANDARDIZE-APART**(r) = ($p_1 \wedge \ldots \wedge p_n \Rightarrow q$)
and UNIFY(*q*, *q′*) succeeds **do**
new_goals ← [p_1', …, p_n', REST(*goals*)]
answers ← FOL-BC-ASK (*KB*,*new_goals*,COMPOSE(*θ′*,*θ*)) ∪ *answers*
return *answers*

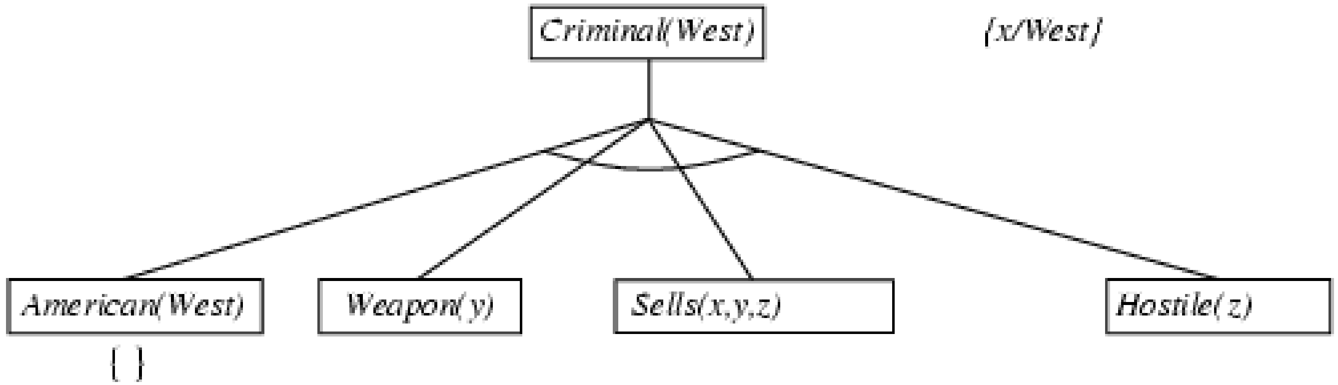
Backward chaining example

$\boxed{Criminal(West)}$

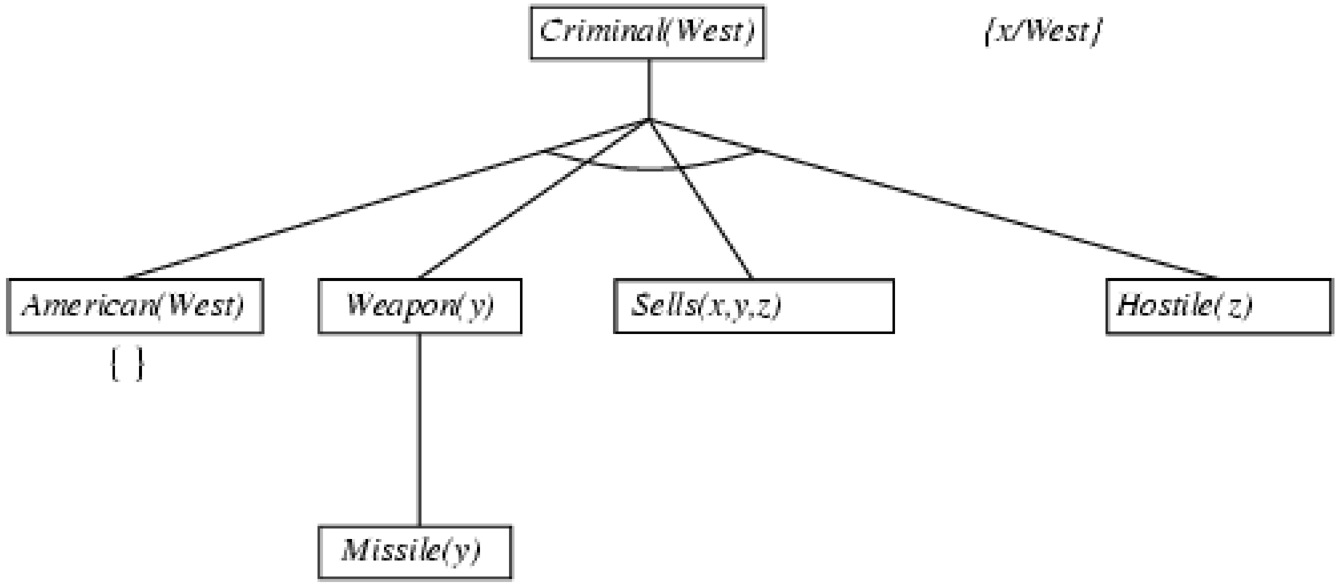
Backward chaining example



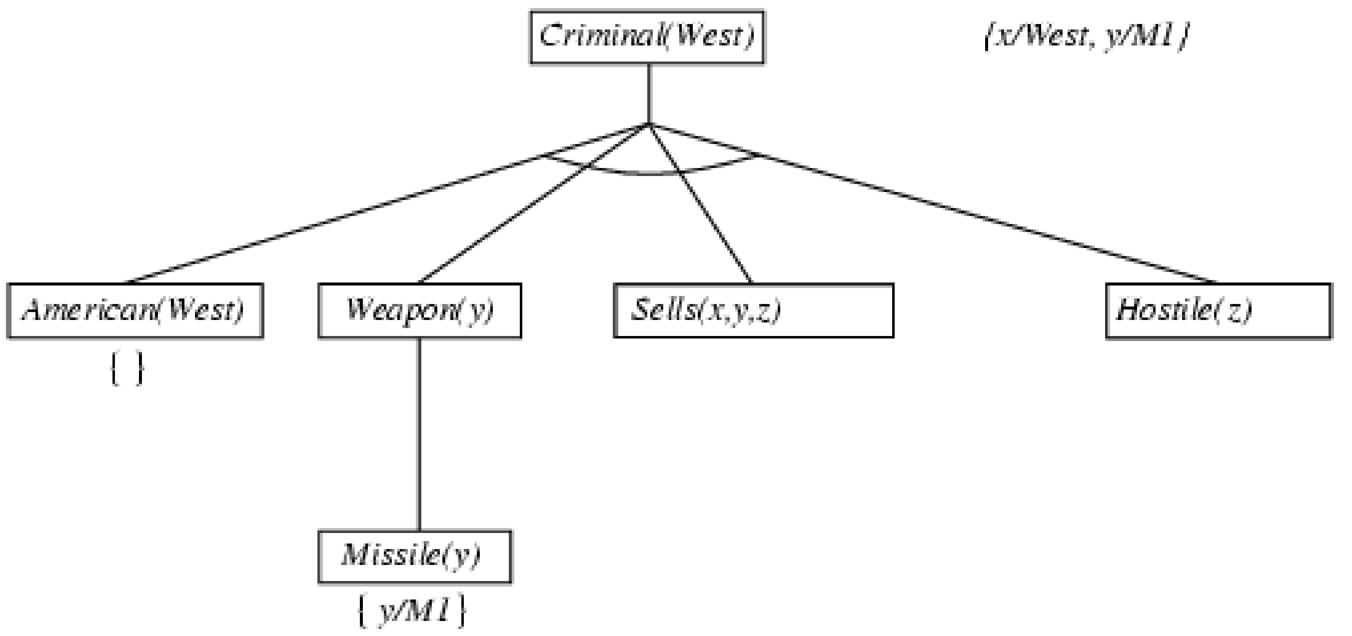
Backward chaining example



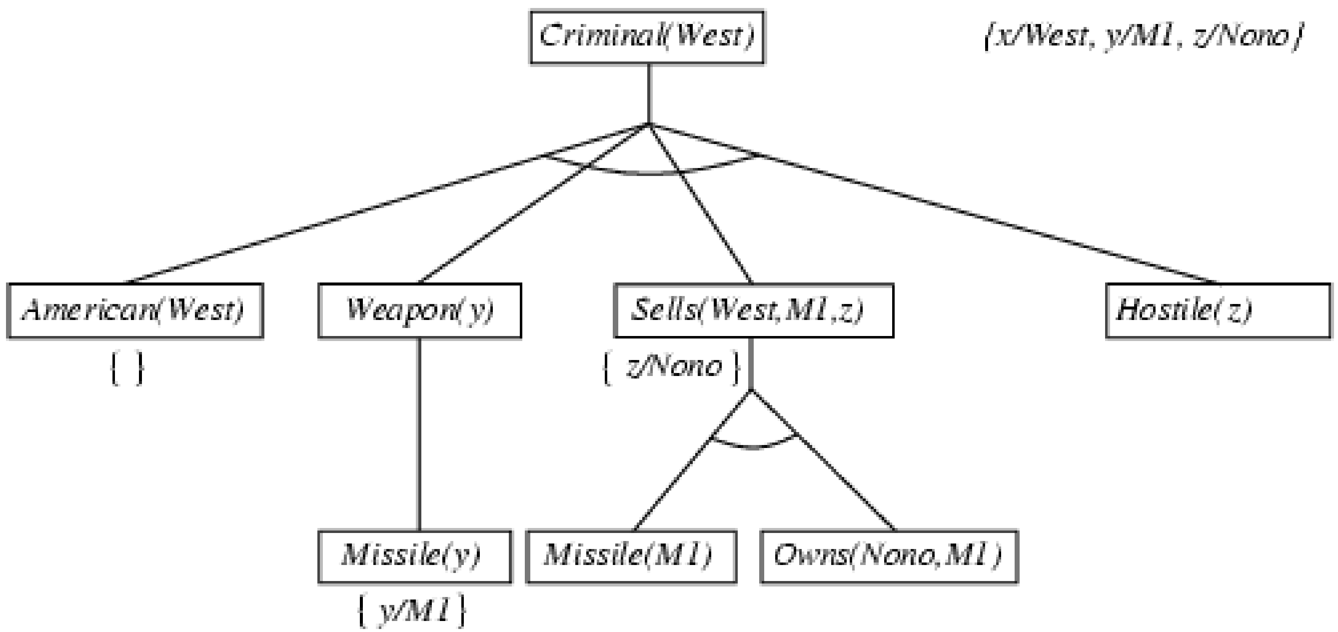
Backward chaining example



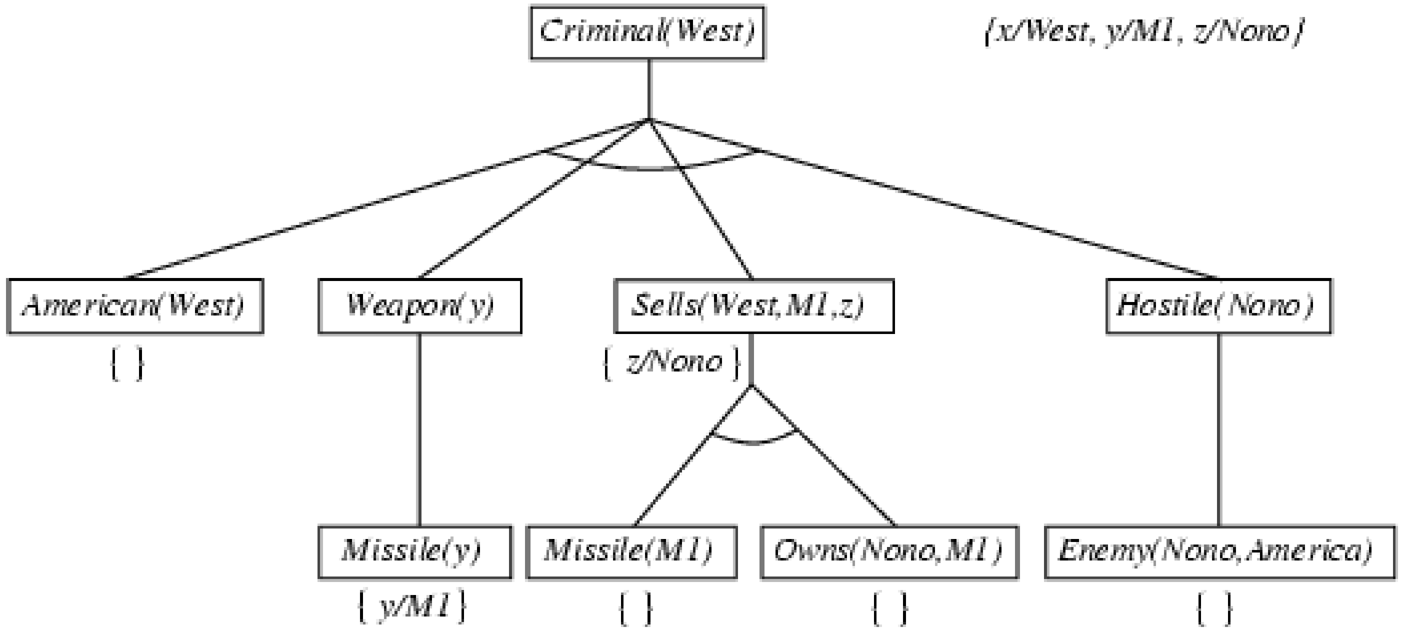
Backward chaining example



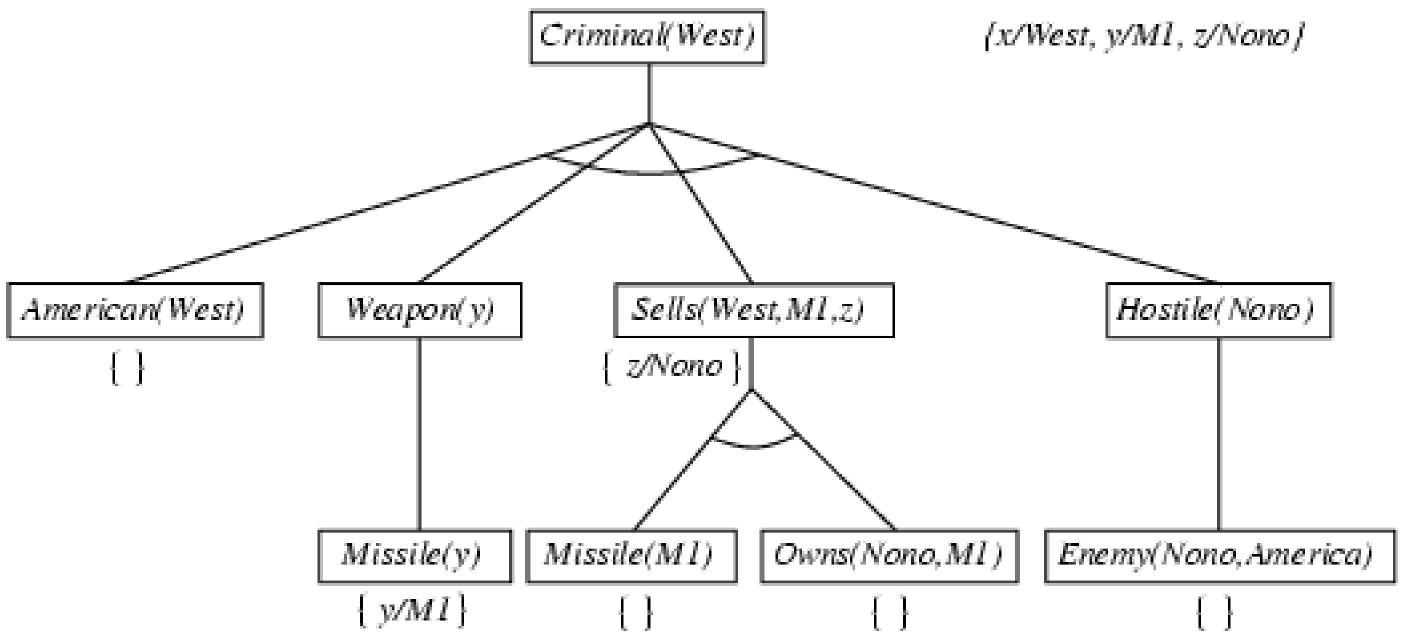
Backward chaining example



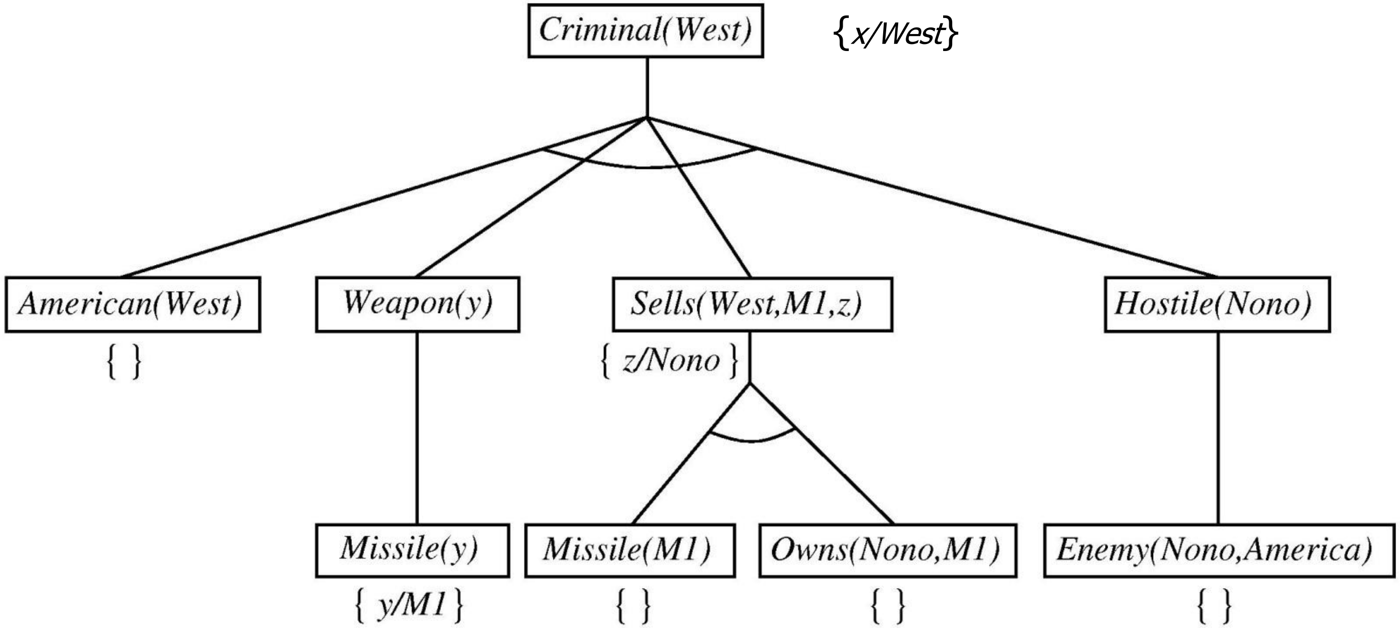
Backward chaining example



Backward chaining example



Backward Chaining Proof Tree



The tree should be read depth-first, left-to-right

Note: once one sub-goal in a conjunction succeeds, its substitution is applied to subsequent goals.

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 sub-goals (both success and failure)
  - $\Rightarrow$ fix using caching of previous results (extra space)
- Widely used for logic programming

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

- *The two clauses are assumed to be standardized apart so that they share no variables.*
- For example,

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

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

Apply resolution steps to CNF(KB $\wedge$ $\neg\alpha$); complete for FOL

Conjunctive Normal Form in FOL

- Resolution requires sentences to be in conjunctive normal form (CNF) = *a conjunction of clauses, where each clause is a disjunction of literals*
  - Literals can contain variables, which are assumed to be universally quantified.

Example: for each child there is a toy and a park where he can play happily!

$$\forall x\, [Child(x) \wedge\ \exists \mathrm{y}\ \mathrm{Toy}(y) \wedge\ \exists z Park(z) \wedge \mathrm{Plays}(x, y, z) \Rightarrow\ Happy(x)]$$

becomes transformed in CNF:

$$\boxed{\neg\mathrm{Child}(x)\ \vee\ \neg\mathrm{Toy}(y)\ \vee \neg\mathrm{Park}(z) \vee \neg\mathrm{Plays}(x, y, \mathrm{z}) \vee\ Happy(x)}$$

Conversion to Normal Form

- *Every sentence of first-order logic can be converted into an inferentially equivalent CNF sentence.*
- *<u>Illustrate the procedure by translating the sentence:</u>*

"Everyone who loves all animals is loved by someone"

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

The Steps

- **Step 1** Eliminate implications:

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

- **Step 2** Move **¬** inwards
  - In addition to the usual rules for negated connectives, we need rules for negated quantifiers:

$\neg \forall x\ p$ becomes $\exists x \neg p$ $\qquad$ $\neg \exists x\ p$ becomes $\forall x \neg p$

  - Our sentence goes through the following transformations:

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

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

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

<u>Step 3</u> *Standardize Variables*

- For sentences like

$$\boxed{(\forall x\ \mathrm{P}(x)) \vee (\exists x\ \mathrm{Q}(x))}$$

which use the same variable twice, change the name of one of the variables.

- *This avoids confusion later when we drop the quantifiers !!!!*

> We have:

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

Step 4 *Skolemize (I)*

- *Skolemization is the process of removing existential quantifiers by elimination.*

In a way it is like the Existential Instatiation rule – translate $\exists x\, P(x)$ into *P(A)* where A is a new constant.

- *Our sentence was:*

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

- *Our sentence becomes:*

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

Interpretation: *everyone either fails to love a particular animal A, or is loved by some particular entity B*

Step 4 *Skolemize II*

Now we have:

$$\boxed{\forall x\, [\text{Animal}(A) \wedge \neg \text{Loves}(x, A)] \vee \text{Loves}(B, x)}$$

- Interpretation: *everyone either fails to love a particular animal A, or is loved by some particular entity B*

The original sentence "*Everyone who loves all animals is loved by someone*" allows each person to fail to love a different animal or to be loved by a different person.

- *Thus we want the Skolem functions to depend on x*:

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

F and G are Skolem functions.

- General rule: *arguments of the Skolem function are all the universally quantified variables in whose scope the existential quantifier(s) appear(s)*.

Step 4 *Skolemize III*

General rule: *arguments of the Skolem function(s) are all the universally quantified variables in whose scope the variable(s) of the existential quantifier(s) appear(s).*

We had:

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

universally quantified variable *+ its scope*

existential quantifiers *2 skolem functs*

- *Thus we want the Skolem functions to depend on x* :

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

F and G are Skolem functions.

Step 5 & Step 6

- **Step 5** Drop universal quantifiers:
  - *At this point, all variables must be universally quantified. Moreover, the sentence is equivalent to one in which all the universal quantifiers have been moved to the left.*
  - We can therefore drop the universal quantifiers:

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

- **Step 6** Distribute $\wedge$ over $\vee$:

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

Sentence in CNF

The Resolution Inference Rule

- *Two clauses, which are assumed to be standardized apart, so that they share no variables, can be resolved if they contain complementary literals (one literal is the negation of the other).*
- We have:

$$\frac{l_1 \vee \ldots \vee l_k, \quad m_1 \vee \ldots \vee m_n}{SUBST(\theta, l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n)}$$

$$\boxed{where\ \theta = UNIFY(l_i, \neg m_j)}$$

- *For example, we can resolve the two clauses:*

$$\boxed{[\mathrm{Animal}(\mathrm{F}(x)) \vee \mathrm{Loves}(\mathrm{G}(x), x)]\ \mathrm{and}\ [\neg \mathrm{Loves}(u, v) \vee \neg \mathrm{Kills}(u, v)]}$$

by eliminating the complementary literals Loves(G(x), x) and ¬Loves(G(x), x) with unifier θ={u/G(x), v/x} to produce the resolvent clause:

$$\boxed{[\mathrm{Animal}(\mathrm{F}(x)) \vee \neg \mathrm{Kills}(\mathrm{G}(x), x)]}$$

Example 1: Proof by Resolution

- <u>Sentences in CNF:</u>

S1: $\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Criminal(x)$

S2: $\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$

S3: $\neg Enemy(x, America) \vee Hostile(x)$

S4: $\neg Missile(x) \vee Weapon(x)$

S5: $Owns(Nono, M_1)$

S6: $Missile(M_1)$

S7: $American(West)$

S8: $Enemy(Nono, America)$

Additionally include the negated goal:

S9: $\neg Criminal(West)$

Resolution

S1: $\neg$American(x) $\vee$ $\neg$Weapon(y) $\vee$ $\neg$Sells(x, y, z) $\vee$ $\neg$Hostile(z) $\vee$ Criminal(x)
S2: $\neg$Missile(x) $\vee$ $\neg$Owns(Nono, x) $\vee$ Sells(West, x, Nono)
S3: $\neg$Enemy(x, America) $\vee$ Hostile(x)
S4: $\neg$Missile(x) $\vee$ Weapon(x)
S5: Owns(Nono, M1)
S6: Missile(M1)
S7: American(West)
S8: Enemy(Nono, America)
S9: $\neg$Criminal(West)

S10: Hostile(Nono) -*from S3 and S8 and SUBST(Nono/x*):
S11: $\neg$American(x) $\vee$ $\neg$Weapon(y) $\vee$ $\neg$Sells(x, y, *Nono*) $\vee$ Criminal(x) *from S10 and S1 and SUBST(Nono/z*):
S12: Weapon(*M1*) -*from S6 and S4 and SUBST(M1/x*):
S13: $\neg$American(x) $\vee$ $\neg$Sells(x, *M1*, *Nono*) $\vee$ Criminal(x) -*from S11 and S12 and SUBST(M1/y*):
S14: $\neg$American(*West*) $\vee$ $\neg$Sells(*West*, *M1*, *Nono*) -*from S13 and S9 and SUBST(West/x*):

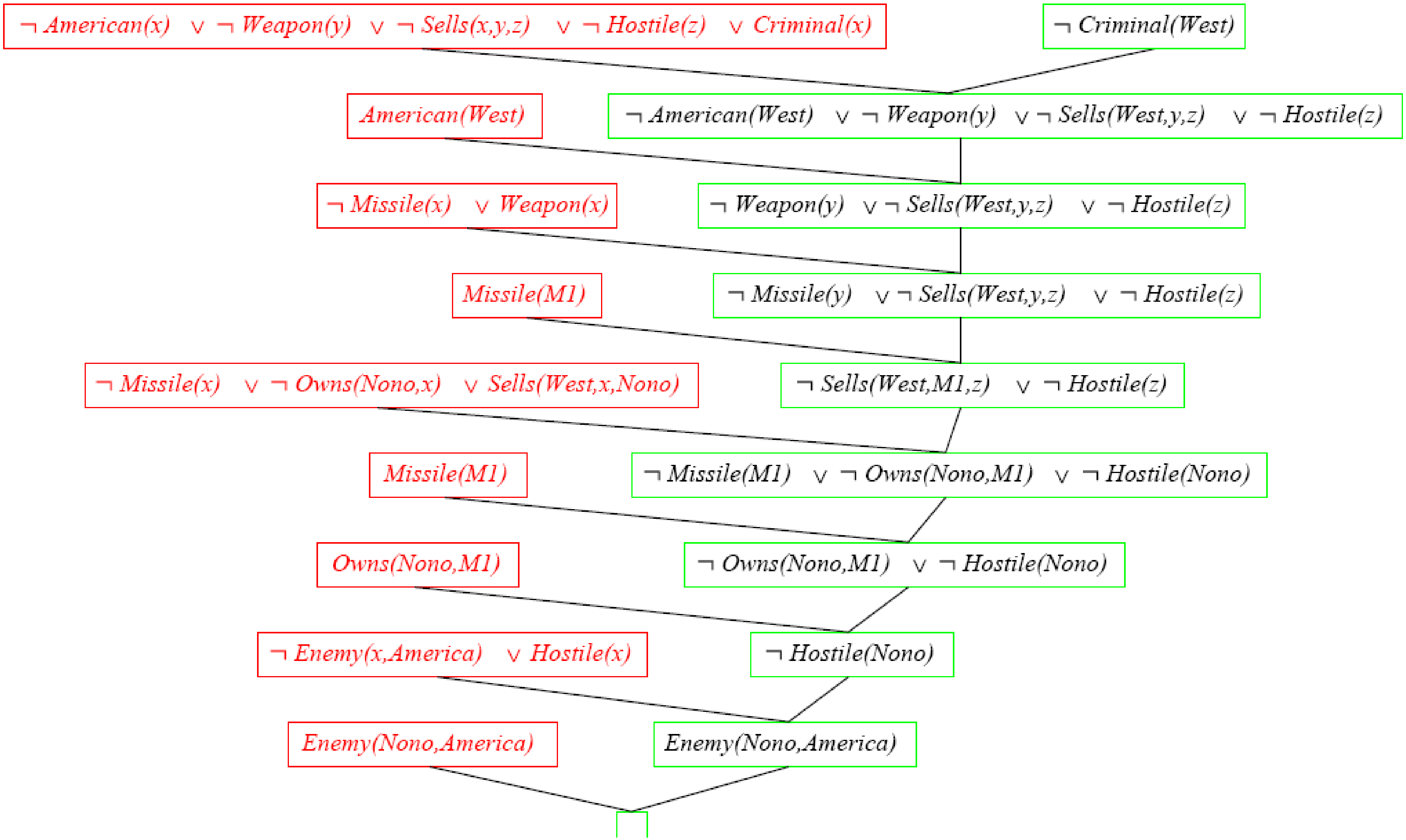
Resolution

S1: ¬American(x) ∨ ¬Weapon(y) ∨ ¬Sells(x, y, z) ∨ ¬Hostile(z) ∨ Criminal(x)
S2: ¬Missile(x) ∨ ¬Owns(Nono, x) ∨ Sells(West, x, Nono)
S3: ¬Enemy(x, America) ∨ Hostile(x)
S4: ¬Missile(x) ∨ Weapon(x)
S5: Owns(Nono, M1)
S6: Missile(M1)
S7: American(West)
S8: Enemy(Nono, America)
S9: ¬Criminal(West)

....
S14: ¬American(*West*) ∨ ¬Sells(*West, M1, Nono*) -*from S13 and S9 and SUBST(West/x*):
S15: ¬Sells(*West, M1, Nono*) -*from S14 and S7*
S16: ¬Owns(Nono, *M1*) ∨ Sells(West, *M1*, Nono) -*from S6 and S2 and SUBST(M1/x*)

S17: Sells(*West, M1, Nono*) -*from S16 and S5*
S18: FALSE from S17 and S15

Resolution proof: definite clauses



More Examples

- In English:
  1. Marcus was a man $\Rightarrow$ man(Marcus)
  2. Marcus was a Pompeian $\Rightarrow$ pompeian(Marcus)
  3. All Pompeians were Romans $\Rightarrow$
     $\forall x$ pompeian(x) $\Rightarrow$ roman(x)
  4. Caesar was a ruler $\Rightarrow$ ruler(Caesar)
  5. All Romans were loyal to Caesar or hated him $\Rightarrow$
     $\forall x$ roman(x) $\Rightarrow$ loyal_to(x, Caesar) $\vee$ hate(x, Caesar)

More Examples

6. Everyone is loyal to someone $\Rightarrow$

$$\forall x \; \exists y \; loyal\_to(x,y)$$

7. People only try to assasinate rulers they are not loyal to $\Rightarrow$

$$\forall x \; \forall y \; (man(x) \wedge ruler(y) \wedge try\_assassinate(x,y) \Rightarrow \neg loyal\_to(x,y)$$

8. Marcus tried to assassinate Caesar $\Rightarrow$

$$try\_assassinate(Marcus, Caesar)$$

9. Did Marcus hate Caesar?

$Hate(Marcus, Caesar)$ $\leftarrow$ what we want to prove!

$$\Downarrow$$

$$\neg Hate(Marcus,Caesar)$$

Convert to Clause Form

1. man(Marcus)
2. pompeian(Marcus)
3. $\forall x_1\ (\neg pompeian(x_1) \vee roman(x_1))$
4. ruler(Caesar)
5. $\forall x_2\ (\neg roman(x_2) \vee loyal\_to(x_2, Caesar) \vee hate(x_2, Caesar))$

   $\neg a \vee (b \vee c) = \neg a \vee b \vee c$
6. $\forall x_3\ loyal\_to(x_3, f_1(x_3))$
7. $\forall x_4\ (\neg man(x_4) \vee \neg ruler(y_1) \vee \neg try\_assassinate(x_4, y_1) \vee$
   $\neg loyal\_to(x_4, y_1))$
8. try_assassinate(Marcus, Caesar)
9. $\neg$hate(Marcus, Caesar)

Resolution

9 & 5 $\Rightarrow$ 10. $\neg$roman(Marcus) $\vee$ loyal_to(Marcus, Caesar)   Subst(x_2/Marcus)

3 & 10 $\Rightarrow$ 11. $\neg$pompeian(Marcus) $\vee$ loyal_to(Marcus,Caesar)
Subst(x_1/Marcus)

2 & 11 $\Rightarrow$ 12. loyal_to(Marcus, Caesar)

7 & 12 $\Rightarrow$ 13. $\neg$man(Marcus) $\vee$ $\neg$ruler(Caesar) $\vee$
$\neg$try_assassinate(Marcus, Caesar)
Subst(x_4/Marcus; y_1/Caesar)

13 & 1 $\Rightarrow$ 14. $\neg$ruler(Caesar) $\vee$ $\neg$tryassasinate(Marcus, Caesar)

14 & 5 $\Rightarrow$ 15. $\neg$tryassasinate(Marcus,Caesar)

15 & 8 $\Rightarrow$ FALSE

Answer Extraction

1. Marcus was a Pompeian $\Rightarrow$ pompeian(Marcus)
2. All Pompeians died at 79 $\Rightarrow$

   $\forall$x pompeian(x) $\Rightarrow$ died(x,79)
3. When did Marcus die??? $\Rightarrow$

   $\exists$y died(Marcus,y)

Goal: Answer the question:

$\neg\exists$y died(Marcus,y)

Resolution

1. Pompeian(Marcus)
2. $\neg$Pompeian(x) $\vee$ died(x,79)
3. $\forall$y $\neg$died(Marcus,y)

$\boxed{\vee \text{ died(Marcus,y)}}$

you must append the negation of the question!

From 2 & 3 $\Rightarrow$ 4. $\neg$Pompeian(Marcus) (if not appended negation! $\Rightarrow$ 4′. $\neg$Pompeian(Marcus) $\vee$ died(Marcus,79)

Subst(x/Marcus; y/79)

From 1 & 4 $\Rightarrow$ NIL

$\Rightarrow$ Died(Marcus,79) if Appended negation of the goal (from 1 & 4′)