UIT2504
Artificial Intelligence

Inferences in First-Order Logic

Outline

- Reducing first-order inference to propositional inference
- Unification
- Conjunctive Normal Form
- Generalized Modus Ponens
- Forward chaining
- Backward chaining
- Resolution

Predicate Logic . . .

Semantics

- Interpretations, models
- Satisfiable, inconsistent, valid, invalid
- Logical consequence

How syntax-based derivations be carried out to perform inferences in FOL?

FOL to PL

First order inference can be done by converting the knowledge base to PL and using propositional inference.

- How to convert universal quantifiers?
  - Replace variable by a ground term.
- How to convert existential quantifiers?
  - Skolemization.

Universal instantiation (UI)

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

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

for any variable *v* and ground term *g*

E.g., $\forall x\ King(x) \land Greedy(x) \Rightarrow Evil(x)$ yields:

$King(John) \land Greedy(John) \Rightarrow Evil(John)$
$King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)$
$King(Father(John)) \land Greedy(Father(John)) \Rightarrow Evil(Father(John))$
.
.
This could translate into infinite sentences in the presence of function symbols!!!

Existential instantiation (EI)

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

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

E.g., $\exists x\ Crown(x) \wedge OnHead(x, John)$ yields:

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

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

We will discuss more about Skolemization later . . .

EI versus UI

UI can be applied several times to *add* new sentences; the new KB is logically equivalent to the old.

EI can be applied only once to replace the existential sentence; the new KB is not equivalent to the old but is satisfiable if the old KB was satisfiable.

Reduction to propositional inference

Suppose the KB contains just the following:

$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
King(John)
Greedy(John)
Brother(Richard, John)

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

Reduction to propositional inference

Suppose the KB contains just the following:

$\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)
King(John)
Greedy(John)
Brother(Richard, John)

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

King(John) $\wedge$ Greedy(John) $\Rightarrow$ Evil(John)
King(Richard) $\wedge$ Greedy(Richard) $\Rightarrow$ Evil(Richard)
King(John)
Greedy(John)
Brother(Richard, John)

The new KB is propositionalized: proposition symbols are

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

Reduction contd.

CLAIM: A ground sentence is entailed by a new KB iff it is entailed by the original KB.

Reduction contd.

CLAIM: A ground sentence is entailed by a new KB iff it is entailed by the original KB.

CLAIM: Every FOL KB can be propositionalized so as to preserve entailment

Reduction contd.

CLAIM: A ground sentence is entailed by a new KB iff it is entailed by the original KB.

CLAIM: Every FOL KB can be propositionalized so as to preserve entailment

IDEA: propositionalize KB and query, apply resolution, return result

Reduction contd.

CLAIM: A ground sentence is entailed by a new KB iff it is entailed by the original KB.

CLAIM: Every FOL KB can be propositionalized so as to preserve entailment

IDEA: propositionalize KB and query, apply resolution, return result

PROBLEM: with function symbols, there are infinitely many ground terms,

e.g., *Father*(*Father*(*Father*(*John*)))

Reduction contd.

THEOREM: Herbrand (1930). If a sentence α is entailed by an FOL KB, it is entailed by a finite subset of the propositionalized KB

Reduction contd.

THEOREM: Herbrand (1930). If a sentence α is entailed by an FOL KB, it is entailed by a finite subset of the propositionalized KB

IDEA: For n = 0 to ∞ do

– create a propositional KB by instantiating with depth-n terms

– see if α is entailed by this KB

Reduction contd.

THEOREM: Herbrand (1930). If a sentence α is entailed by an FOL KB, it is entailed by a finite subset of the propositionalized KB

IDEA: For $n = 0$ to ∞ do

- create a propositional KB by instantiating with depth-n terms
- see if α is entailed by this KB

PROBLEM: works if α is entailed, loops if α is not entailed

Reduction contd.

THEOREM: Herbrand (1930). If a sentence α is entailed by an FOL KB, it is entailed by a finite subset of the propositionalized KB

IDEA: For $n = 0$ to ∞ do

- create a propositional KB by instantiating with depth-n terms
- see if α is entailed by this KB

PROBLEM: works if α is entailed, loops if α is not entailed

THEOREM: Turing (1936), Church (1936) Entailment for FOL is semi decidable

- algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every non-entailed sentence.

Problems with propositionalization

Propositionalization seems to generate lots of irrelevant sentences.

– E.g., from:

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

$King(John)$

$\forall y\ Greedy(y)$

$Brother(Richard, John)$

It seems obvious that *Evil*(*John*) can be inferred, but propositionalization produces lots of facts such as *Greedy*(*Richard*) that are irrelevant.

Problems with propositionalization

Propositionalization seems to generate lots of irrelevant sentences.

–E.g., from:

$\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)
King(John)
$\forall y$ Greedy(y)
Brother(Richard,John)

It seems obvious that *Evil*(*John*), but propositionalization produces lots of facts such as *Greedy*(*Richard*) that are irrelevant.

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

Lifting and Unification

Instead of translating the knowledge base to PL, we can redefine the inference rules into FOL.

– Lifting; they only make those substitutions that are required to allow particular inferences to proceed.

– E.g. *generalized Modus Ponens*

- – To introduce substitutions, different logical expressions have to be look identical
- – 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)*

θ = {x/John, y/John} works

Unify(α, β) = θ if αθ = βθ

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)*

θ = {x/John,y/John} works

Unify(α, β) = θ if αθ = βθ

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)*

θ = {x/John,y/John} works

Unify(α, β) = θ if αθ = βθ

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)*

θ = {x/John,y/John} works

Unify(α, β) = θ if αθ = βθ

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)*

θ = {x/John,y/John} works

Unify(α, β) = θ 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

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

Unification

To unify *Knows(John,x)* and *Knows(y,z)*,

$\theta 1$ = {y/John, x/z } or

$\theta 2$ = {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 }

The unification 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

   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(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
```

The unification algorithm

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

  if $\{var/val\} \in \theta$ **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

Conversion to CNF

Everyone who loves all animals is loved by someone:

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

Eliminate bi-conditionals and implications

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

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

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

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

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

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

Conversion to CNF contd.

- Standardize variables: each quantifier should use a different one:

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

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

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

- Drop universal quantifiers:

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

- Distribute $\vee$ over $\wedge$:

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

Generalized Modus Ponens (GMP)

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

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

p_1' is *King*(*John*)
p_2' is *Greedy*(*y*)
θ is {x/John,y/John}
qθ is *Evil*(*John*)

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

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

All variables assumed universally quantified. (part of normalization)

Generalized Modus Ponens (GMP)

$$\frac{King(x) \wedge Greedy(x) \Rightarrow Evil(x) \qquad King(John) \qquad Greedy(y)}{Evil(John)}$$

p_1' is *King*(*John*)
p_2' is *Greedy*(*y*)
θ is {x/John,y/John}
qθ is *Evil*(*John*)

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

GMP can be used in forward chaining mode for bottom-up reasoning or in backward chaining mode for top-down goal focused reasoning

Example knowledge base

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

Prove that Col. West is a criminal

Example knowledge base contd.

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

Example knowledge base contd.

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

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

Example knowledge base contd.

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

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

Nono … has some missiles

Example knowledge base contd.

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

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

Nono … has some missiles, i.e.,

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

$Owns(Nono, M_1)$

$Missile(M_1)$

Example knowledge base contd.

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

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

Nono … has some missiles, i.e.,

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

Owns(Nono, M_1*)*

Missile(M_1*)*

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

Example knowledge base contd.

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

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

Nono … has some missiles, i.e.,

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

Owns(Nono,M_1)

Missile(M_1)

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

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

Example knowledge base contd.

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

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

Nono … has some missiles, i.e., $\exists x$ Owns(Nono,x) $\wedge$ Missile(x):

Owns(Nono,M_1)

Missile(M_1)

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

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

Missiles are weapons:

Example knowledge base contd.

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

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

Nono … has some missiles, i.e., $\exists$x Owns(Nono,x) $\wedge$ Missile(x):

Owns(Nono,M_1)
Missile(M_1)

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

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

Missiles are weapons:

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

Example knowledge base contd.

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

American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)

Nono … has some missiles, i.e., ∃x Owns(Nono,x) ∧ Missile(x):

Owns(Nono,M_1)
Missile(M_1)

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

Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)

Missiles are weapons:

Missile(x) ⇒ Weapon(x)

An enemy of America counts as "hostile":

Example knowledge base contd.

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

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

Nono … has some missiles, i.e., $\exists x\ Owns(Nono,x) \wedge Missile(x)$:

$Owns(Nono,M_1)$

$Missile(M_1)$

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

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

Missiles are weapons:

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

An enemy of America counts as "hostile“:

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

Example knowledge base contd.

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

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

Nono … has some missiles, i.e., $\exists x\ Owns(Nono,x) \wedge Missile(x)$:

$Owns(Nono,M_1)$
$Missile(M_1)$

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

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

Missiles are weapons:

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

An enemy of America counts as "hostile“:

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

West, who is American …

Example knowledge base contd.

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

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

Nono … has some missiles, i.e., $\exists$x Owns(Nono,x) $\wedge$ Missile(x):

Owns(Nono,M_1)

Missile(M_1)

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

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

Missiles are weapons:

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

An enemy of America counts as "hostile":

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

West, who is American …

American(West)

Example knowledge base contd.

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

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

Nono … has some missiles, i.e., $\exists x\ Owns(Nono,x) \wedge Missile(x)$:

$Owns(Nono,M_1)$

$Missile(M_1)$

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

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

Missiles are weapons:

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

An enemy of America counts as "hostile“:

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

West, who is American …

$American(West)$

The country Nono, an enemy of America …

Example knowledge base contd.

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

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

Nono … has some missiles, i.e., $\exists$x Owns(Nono,x) $\wedge$ Missile(x):

Owns(Nono,M_1*)*

Missile(M_1*)*

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

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

Missiles are weapons:

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

An enemy of America counts as "hostile‟:

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

West, who is American …

American(West)

The country Nono, an enemy of America …

Enemy(Nono,America)

Forward chaining algorithm

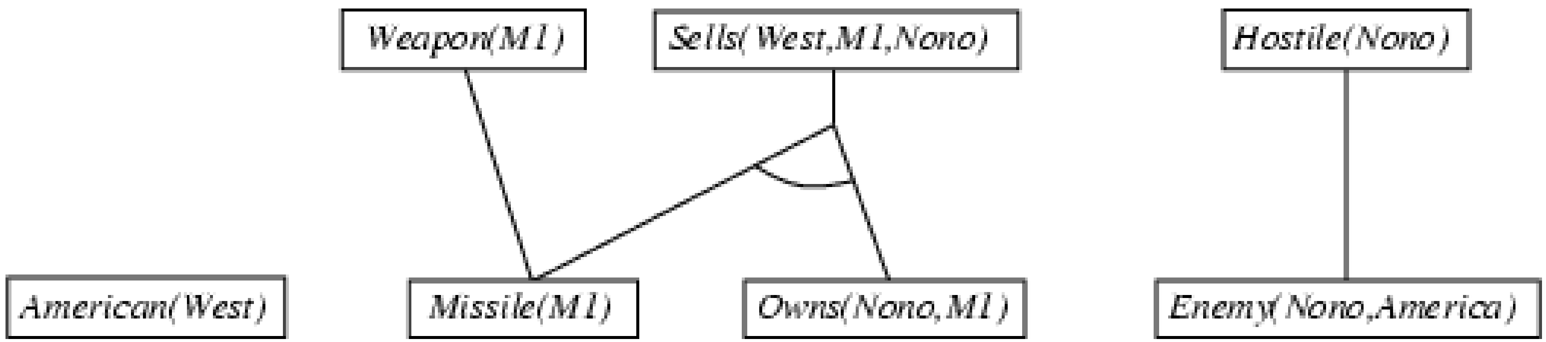
function FOL-FC-ASK(KB, α) **returns** a substitution or *false*

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

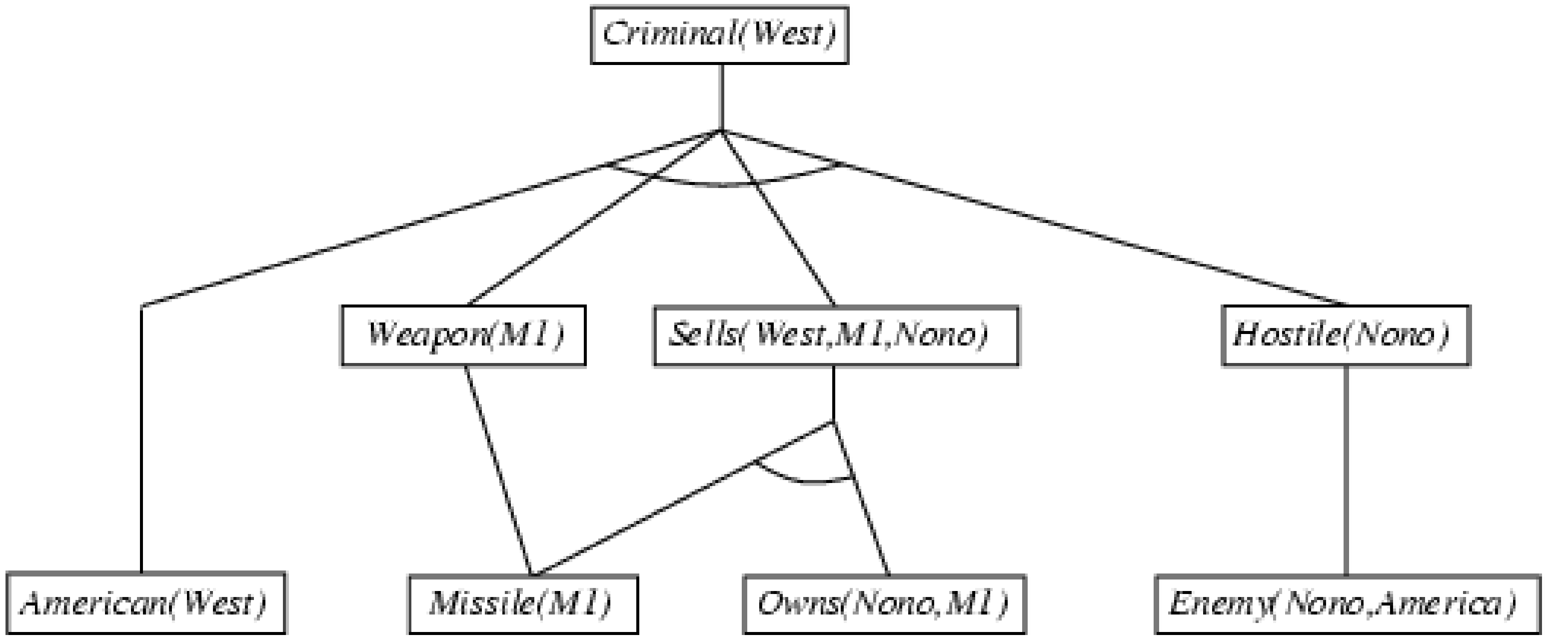
Forward chaining example

$American(West)$ $\quad$ $Missile(M1)$ $\quad$ $Owns(Nono,M1)$ $\quad$ $Enemy(Nono,America)$

Forward chaining example



Forward chaining example



Properties of forward chaining

Sound and complete for first-order definite clauses.

–Cfr. Propositional logic proof.

Datalog = first-order definite clauses + *no functions* (e.g. crime KB)

–FC terminates for Datalog in finite number of iterations

May not terminate in general for DF clauses with functions if α is not entailed

–This is unavoidable: entailment with definite clauses is semi-decidable

Forward chaining is widely used in deductive databases and production systems

Efficiency of forward chaining

- Matching itself can be expensive:
- **Database indexing** allows O(1) retrieval of known facts
- e.g., query *Missile(x)* retrieves *Missile*(M_1)
- *Sells(x,y,z), Sells(West,y,z), Sells(x,Nono,z), Sells(West,Nono,z), etc.*
- (***Subsumption lattice***)
- Matching conjunctive premises against known facts is NP-hard. (*Pattern matching*)
- *Incremental forward chaining*: no need to match a rule on iteration *k* if a premise wasn't added on iteration *k-1*

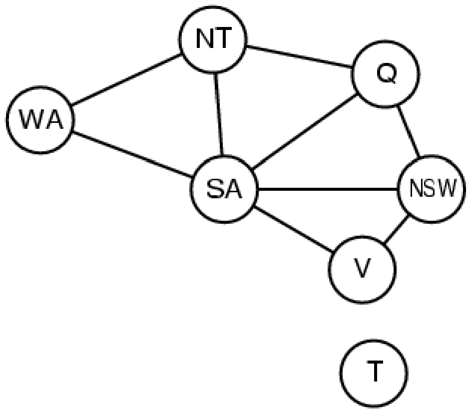
–match each rule whose premise contains a newly added positive literal.(**Rete Algorithm**)

- Forward chaining is not goal directed. May generate many irrelevant sentences. One possible solution: **magic set**

Magic(x) ^ American(x) ^ . . . => Criminal(x)

Magic(West)

Hard matching example



Diff(wa,nt) $\wedge$ *Diff(wa,sa)* $\wedge$ *Diff(nt,q)* $\wedge$
Diff(nt,sa) $\wedge$ *Diff(q,nsw)* $\wedge$ *Diff(q,sa)* $\wedge$
Diff(nsw,v) $\wedge$ *Diff(nsw,sa)* $\wedge$ *Diff(v,sa)* $\Rightarrow$
Colorable()

Diff(Red,Blue) *Diff (Red,Green)*
Diff(Green,Red) *Diff(Green,Blue)*
Diff(Blue,Red) *Diff(Blue,Green)*

Colorable() is inferred iff the CSP has a solution

CSPs include 3SAT as a special case, hence matching is NP-hard

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, the current substitution, initially the empty substitution $\{\,\}$
  local variables: ans, a set of substitutions, initially empty

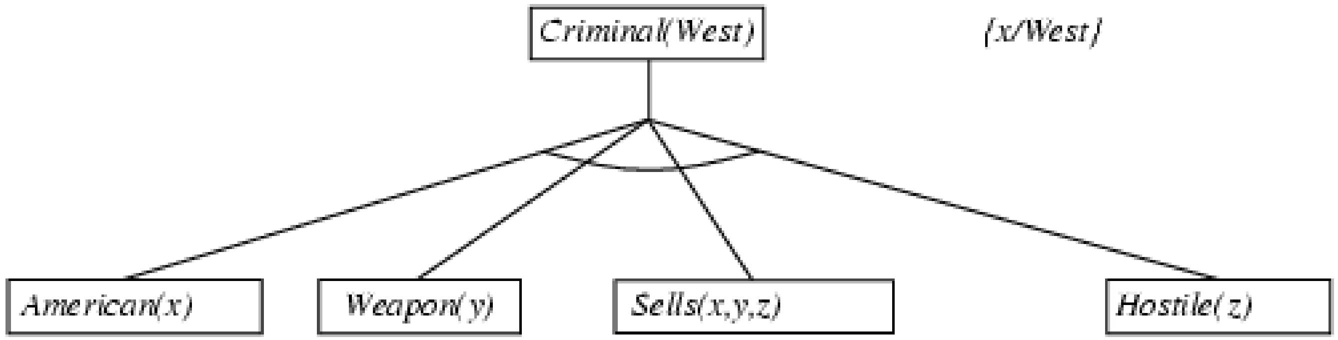
  if $goals$ is empty **then return** $\{\theta\}$
  $q' \leftarrow \textsc{Subst}(\theta, \textsc{First}(goals))$
  for each r **in** KB where $\textsc{Standardize-Apart}(r) = (\, p_1 \ \wedge \ \ldots \ \wedge \ p_n \ \Rightarrow \ q)$
    and $\theta' \leftarrow \textsc{Unify}(q, q')$ succeeds
   $ans \leftarrow \text{FOL-BC-Ask}(KB, [p_1, \ldots, p_n | \textsc{Rest}(goals)], \textsc{Compose}(\theta, \theta')) \cup ans$
  return ans

SUBST(COMPOSE(θ_1, θ_2), p) = SUBST(θ_2, SUBST(θ_1, p))

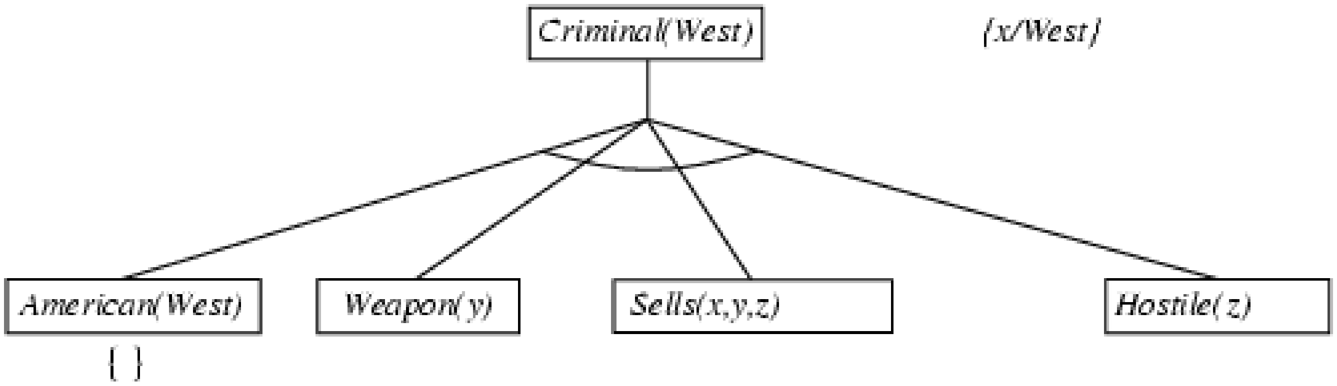
Backward chaining example

$\boxed{Criminal(West)}$

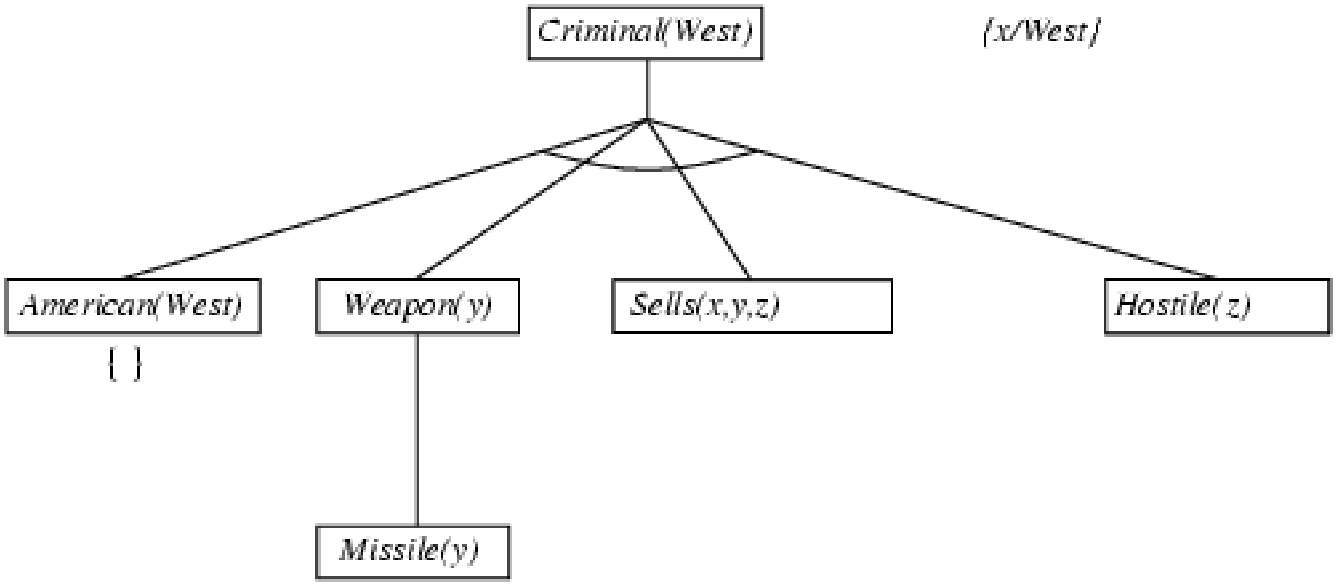
Backward chaining example



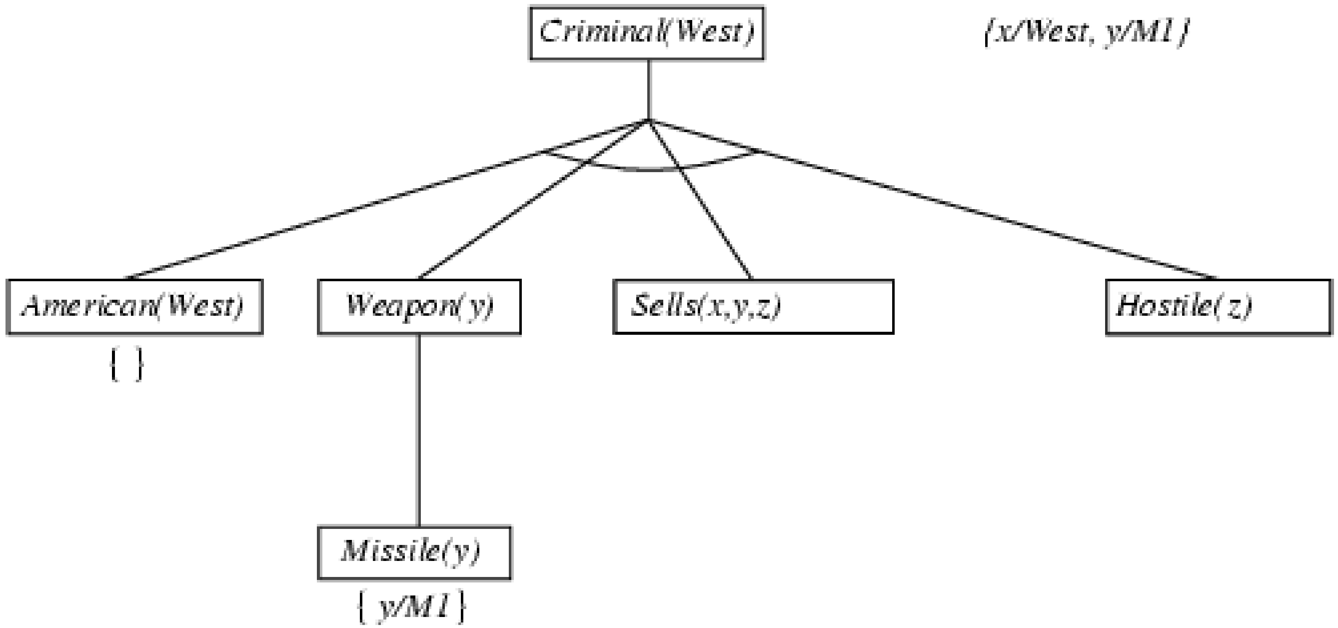
Backward chaining example



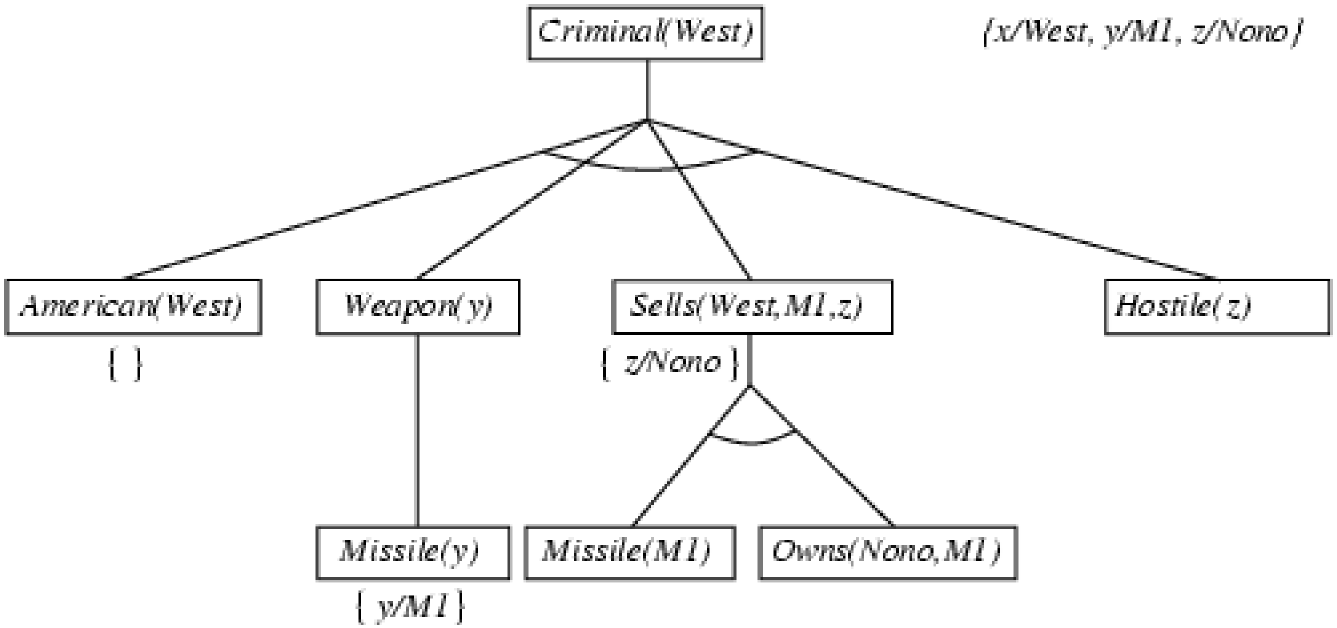
Backward chaining example



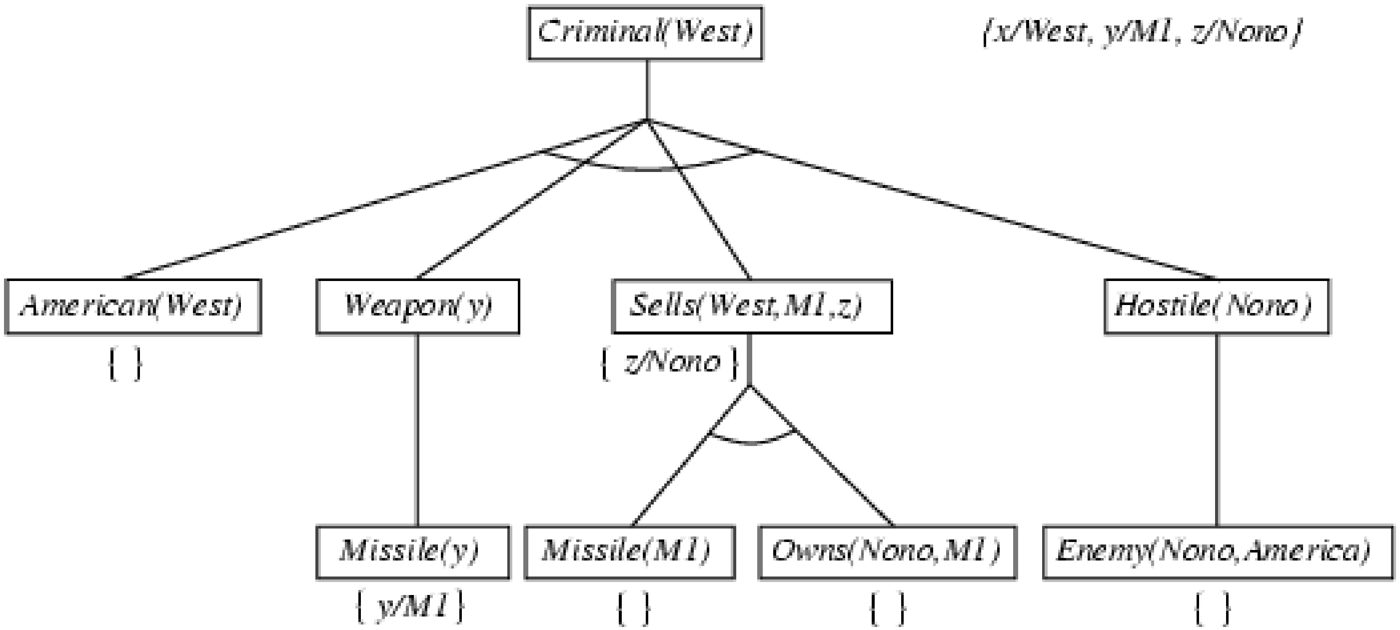
Backward chaining example



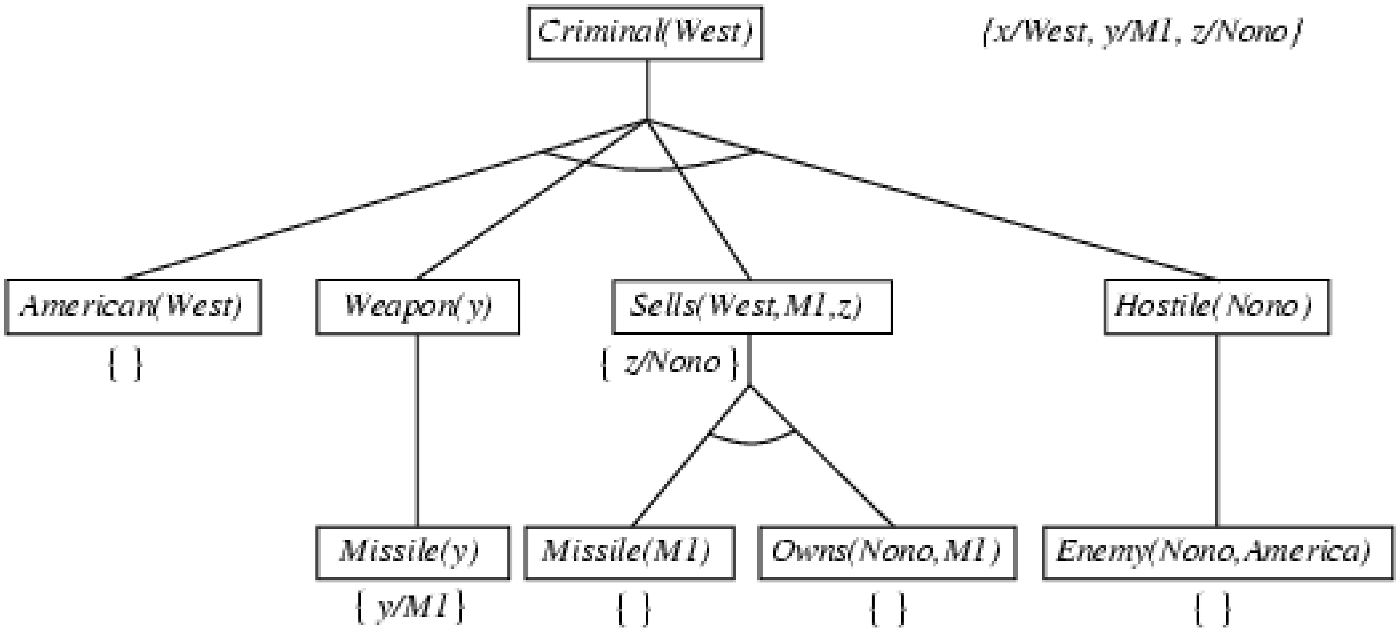
Backward chaining example



Backward chaining example



Backward chaining example



Properties of backward chaining

- Goal-directed
- Depth-first recursive proof search: space is linear in size of proof.
- Incomplete due to infinite loops
  - –fix by checking current goal against every goal on stack
- Inefficient due to repeated sub-goals (both success and failure)
  - –fix using caching of previous results (extra space!!)
- Widely used for logic programming

Resolution: brief summary

Full first-order version:

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

where $\texttt{Unify}(l_i, \neg m_j) = \theta$.

The two clauses are assumed to be standardized apart so that they share no variables.

Two input clauses are resolved together by unifying complementary literals in them resulting in a resolvent

Apply resolution steps to *CNF(KB* $\wedge \neg\alpha$*) and derive an empty clause*

Complete for FOL

Resolution: brief summary

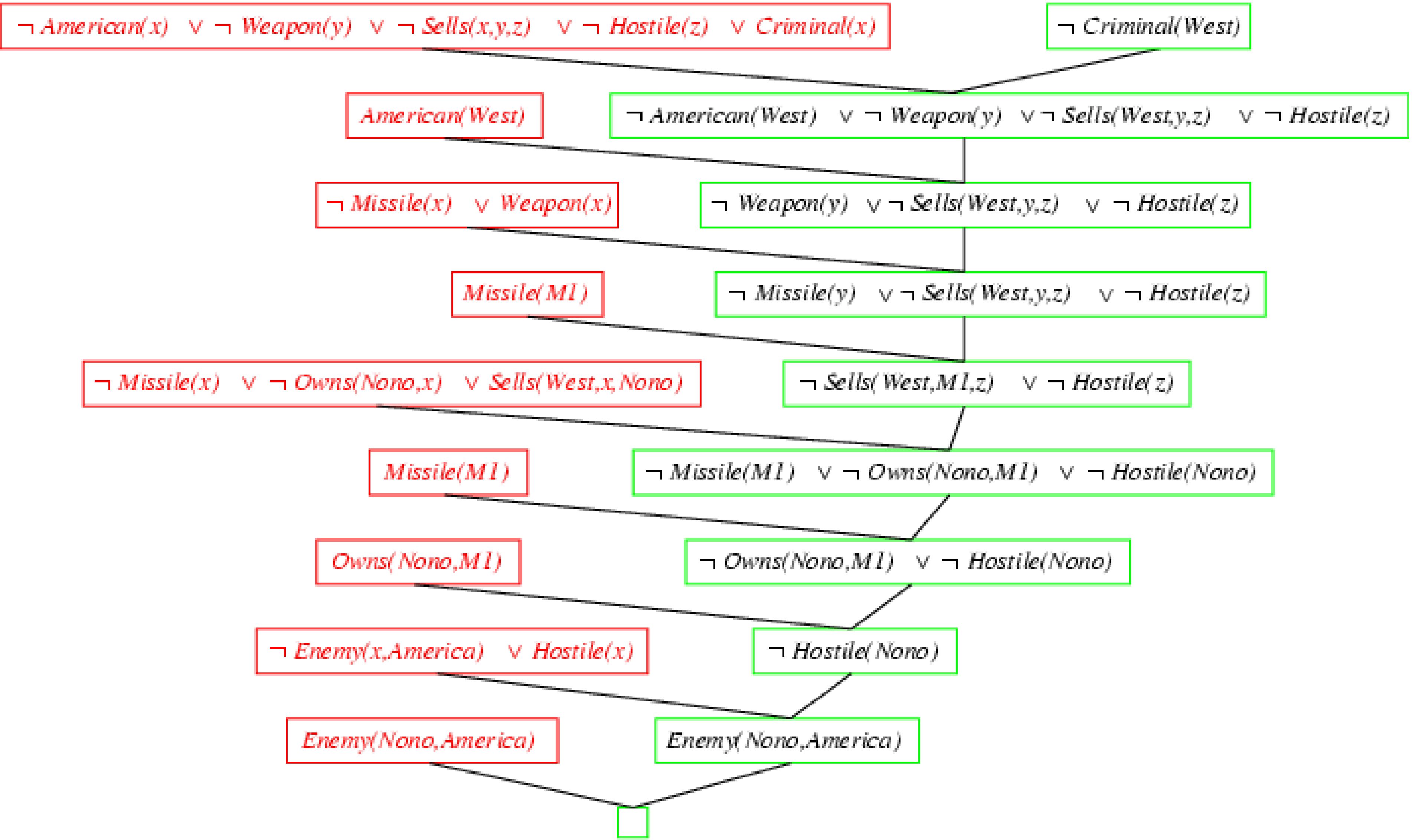
For example,

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

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

Like GMP, resolution can also be used in both forward chaining and backward chaining modes.

Resolution proof: definite clauses



Exercise

Represent in FOL:

– Cats chase mice
– Tom is a cat
– Jerry is a mouse

Use resolution to show that Tom chases Jerry.

Exercise

1. ans(X,Y) :- par(X,Y).
2. ans(X,Y) :- par(Z,Y), ans(X,Z).
3. par(a,b).
4. par(c,b).
5. par(d,e).
6. par(b,e).
7. par(b,f).
8. par(e,g).

Answer the following:

```
?- ans(a,e).
?- ans(a,Who).
?- ans(Who,f).
```

Resolution Strategies

Unit preference
– Prefers one of the input clauses to be a unit clause

Set of Support Strategy
– One of the input clauses should be from a special set and resolvent is added to that special set

Input Resolution
– One of the clauses is from the original KB

Linear Resolution
– Resolvent becomes input clause for next resolution

Linear Input Resolution
– Combines linear and input strategies (and also set-of-support!)

Prolog uses linear input resolution with selection function, referred to as SLD-resolution
– SLDNF refers to SLD resolution with negation as failure

Summary

- Reducing first-order inference to propositional inference
- Unification
- Generalized Modus Ponens
- Forward chaining
- Backward chaining
- Resolution