First-Order Logic
Inference

Reading: Chapter 8, 9.1-9.2, 9.5.1-9.5.5

FOL Syntax and Semantics read: 8.1-8.2

FOL Knowledge Engineering read: 8.3-8.5

FOL Inference read: Chapter 9.1-9.2, 9.5.1-9.5.5

(Please read lecture topic material before and after each lecture on that topic)

Outline

- Reducing first-order inference to propositional inference
- Unification
- Generalized Modus Ponens
- Forward chaining
- Backward chaining
- Resolution
- Other types of reasoning
  - Induction, abduction, analogy
  - Modal logics

You will be expected to know

- Concepts and vocabulary of unification, CNF, and resolution.
- Given two FOL terms containing variables
  - Find the most general unifier if one exists.
  - Else, explain why no unification is possible.
  - See figure 9.1 and surrounding text in your textbook.
- Convert a FOL sentence into Conjunctive Normal Form (CNF).
- Resolve two FOL clauses in CNF to produce their resolvent, including unifying the variables as necessary.
- Produce a short resolution proof from FOL clauses in CNF.

Universal instantiation (UI)

- Notation: Subst({v/g}, α) means the result of substituting ground term g for variable v in sentence α

- 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:

$King(John) \land Greedy(John) \Rightarrow Evil(John)$, $\{x/John\}$

$King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)$, $\{x/Richard\}$

$King(Father(John)) \land Greedy(Father(John)) \Rightarrow Evil(Father(John))$, $\{x/Father(John)\}$

.

Existential instantiation (EI)

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

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

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

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

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

- Existential and universal instantiation allows to "propositionalize" any FOL sentence or KB
  - EI produces one instantiation per EQ sentence
  - UI produces a whole set of instantiated sentences per UQ sentence

Reduction to propositional form

Suppose the KB contains 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: (there are only two ground terms: John and Richard)

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 with "propositions":

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

Reduction continued

- Every FOL KB can be propositionalized so as to preserve entailment
  - A ground sentence is entailed by new KB iff entailed by original KB
  -
- Idea for doing inference in FOL:
  - propositionalize KB and query
  - apply resolution-based inference
  - return result
  -
- Problem: with function symbols, there are infinitely many ground terms,
  - e.g., *Father*(*Father*(*Father*(*John*))), etc

Reduction continued

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

Problem: works if ɑ is entailed, loops if ɑ is not entailed.

→ The problem of semi-decidable: algorithms exist to prove entailment, but no algorithm exists to to prove non-entailment for every non-entailed sentence.

Other Problems with Propositionalization

- Propositionalization generates lots of irrelevant sentences
  - So inference may be very inefficient

- 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*) is entailed, 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

- Lets see if we can do inference directly with FOL sentences

Unification

- Recall: Subst(θ, p) = result of substituting θ into sentence p

- Unify algorithm: takes 2 sentences p and q and returns a unifier if one exists

  Unify(p,q) = θ   where Subst(θ, p) = Subst(θ, q)

- Example:
  p = Knows(John,x)
  q = Knows(John, Jane)

  Unify(p,q) = {x/Jane}

Unification examples

- simple example: query = Knows(John,x), i.e., who does John know?

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}

- Last unification fails: only because x can't take values John and OJ at the same time
  - But we know that if John knows x, and everyone (x) knows OJ, we should be able to infer that John knows OJ
- Problem is due to use of same variable x in both sentences
- Simple solution: Standardizing apart eliminates overlap of variables, e.g., Knows(z,OJ)

Unification

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

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

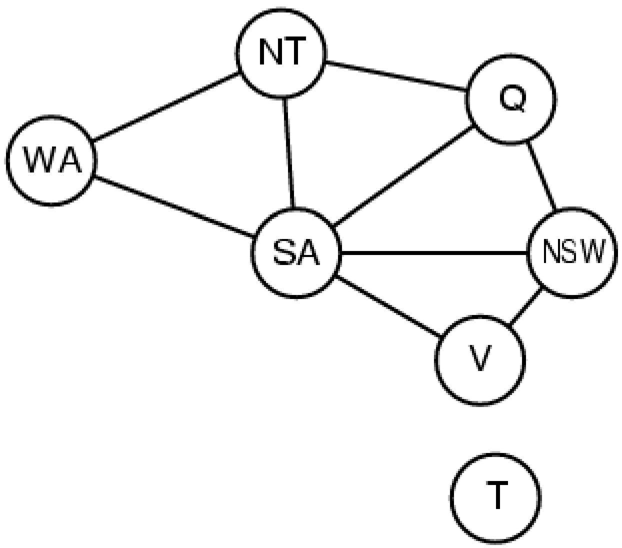
- The first unifier is more general than the second.

- There is a single most general unifier (MGU) that is unique up to renaming of variables.

  MGU = { y/John, x/z }

- General algorithm in Figure 9.1 in the text

Hard matching example



Diff(wa,nt) ∧ *Diff(wa,sa)* ∧ *Diff(nt,q)* ∧
Diff(nt,sa) ∧ *Diff(q,nsw)* ∧ *Diff(q,sa)* ∧
Diff(nsw,v) ∧ *Diff(nsw,sa)* ∧ *Diff(v,sa)* ⇒
Colorable()

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

- To unify the grounded propositions with premises of the implication you need to solve a CSP!
- *Colorable*() is inferred iff the CSP has a solution
- CSPs include 3SAT as a special case, hence matching is NP-hard

Recall our example…

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

And we would like to infer Evil(John) without propositionalization

Generalized Modus Ponens (GMP)

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

where we can unify p_i' and p_i for all *i*

Example:

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

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

θ is {x/John,y/John} q is *Evil*(*x*)

Subst(θ,q) is *Evil*(*John*)

- Implicit assumption that all variables universally quantified

Completeness and Soundness of GMP

- GMP is sound
  - Only derives sentences that are logically entailed
  - See proof in text on p. 326 (3rd ed.; p. 276, 2nd ed.)

- GMP is complete for a KB consisting of definite clauses
  - Complete: derives all sentences that are entailed
  - OR...answers every query whose answers are entailed by such a KB

  - Definite clause: disjunction of literals of which exactly 1 is positive,

    e.g., King(x) AND Greedy(x) -> Evil(x)

    NOT(King(x)) OR NOT(Greedy(x)) OR Evil(x)

Inference appoaches in FOL

- Forward-chaining
  - Uses GMP to add new atomic sentences
  - Useful for systems that make inferences as information streams in
  - Requires KB to be in form of first-order definite clauses

- Backward-chaining
  - Works backwards from a query to try to construct a proof
  - Can suffer from repeated states and incompleteness
  - Useful for query-driven inference

- Resolution-based inference (FOL)
  - Refutation-complete for general KB
    - Can be used to confirm or refute a sentence p (but not to generate all entailed sentences)
  - Requires FOL KB to be reduced to CNF
  - Uses generalized version of propositional inference rule

- Note that all of these methods are generalizations of their propositional equivalents

Knowledge Base 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.
-

Knowledge Base 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.
-

... 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*) and 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)

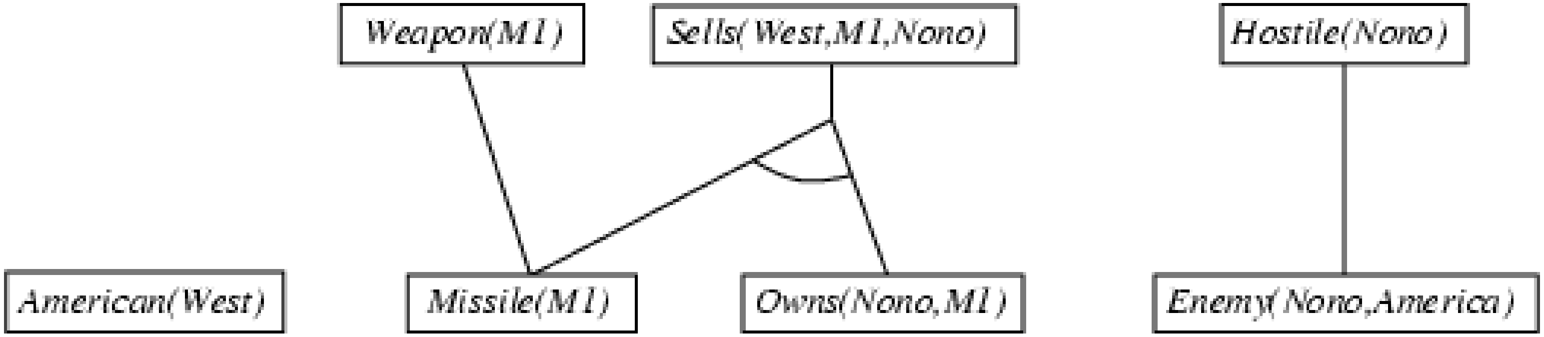
The country Nono, an enemy of America ...

Enemy(Nono,America)

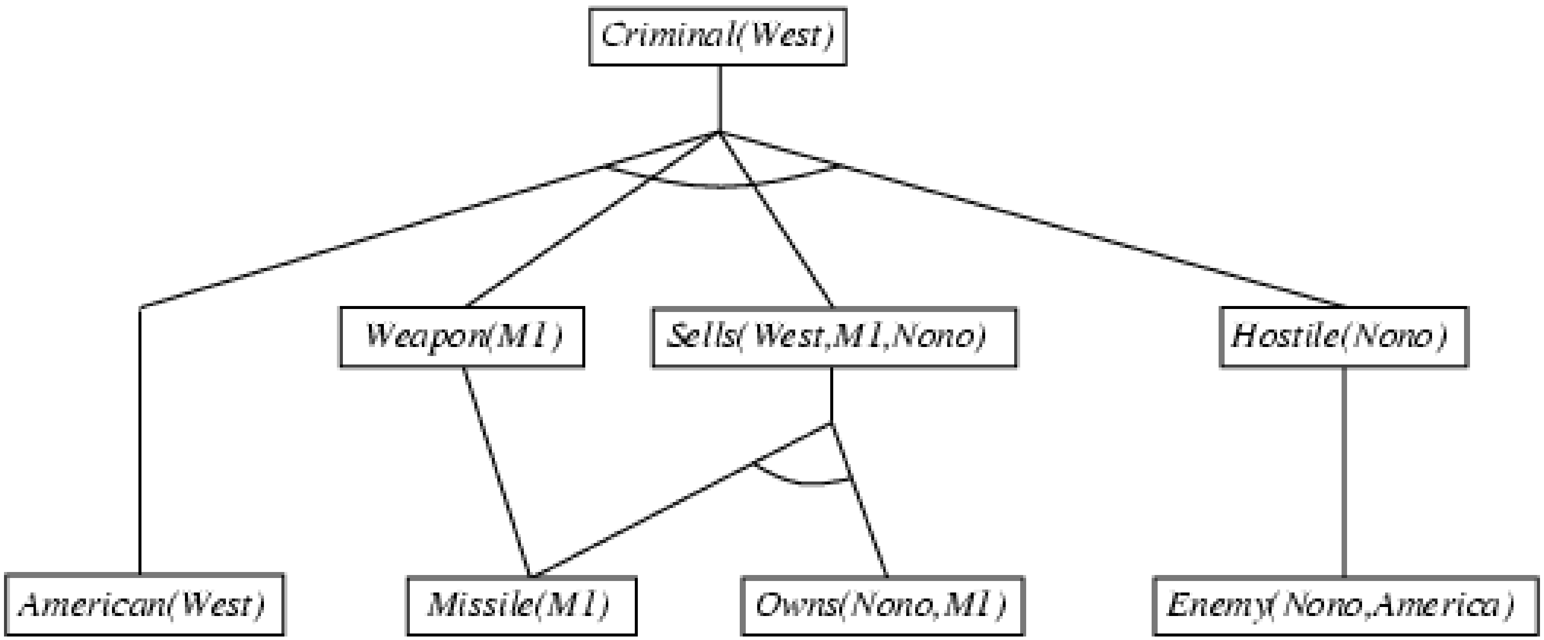
Forward chaining proof

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

Forward chaining proof



Forward chaining proof

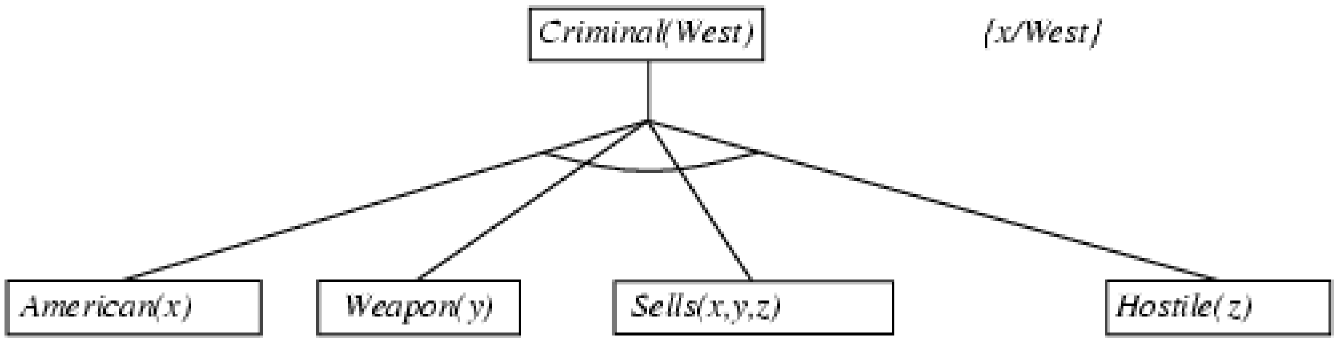


Properties of forward chaining

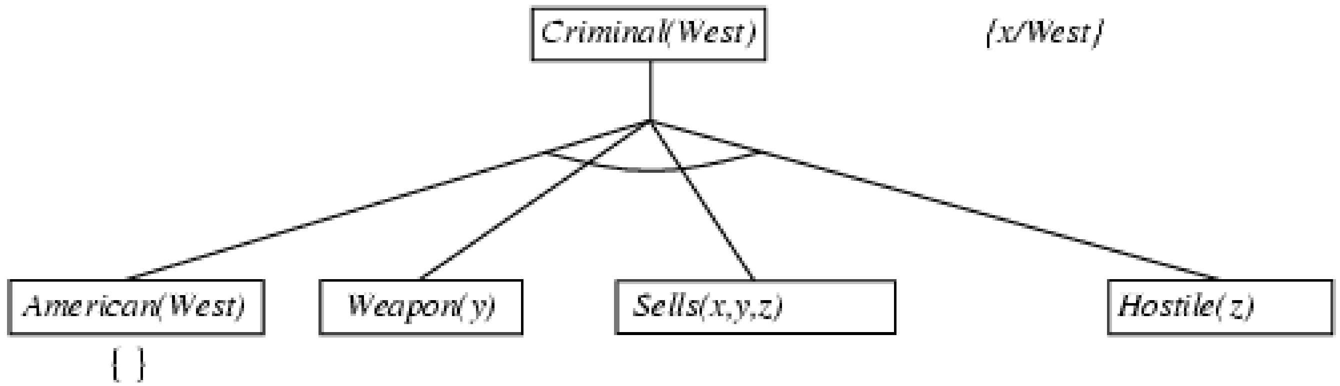
- Sound and complete for first-order definite clauses
- Datalog = first-order definite clauses + no functions
- FC terminates for Datalog in finite number of iterations
- May not terminate in general if α is not entailed
- Incremental forward chaining: no need to match a rule on iteration *k* if a premise wasn't added on iteration *k-1*
  $\Rightarrow$ match each rule whose premise contains a newly added positive literal

Backward chaining example

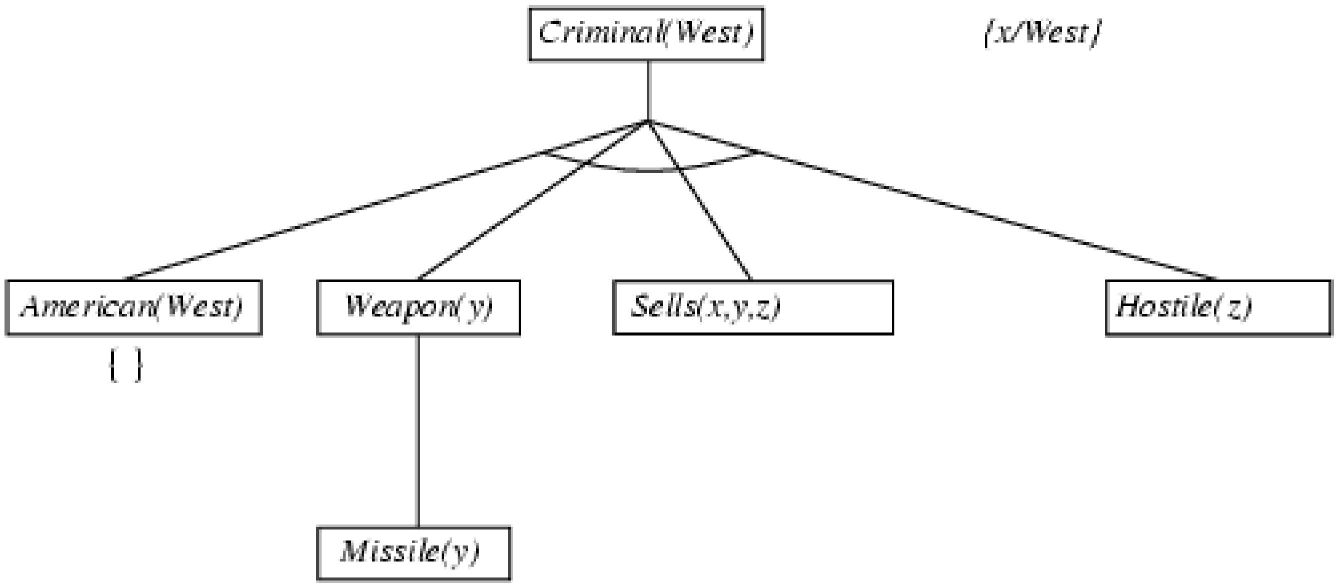
Backward chaining example



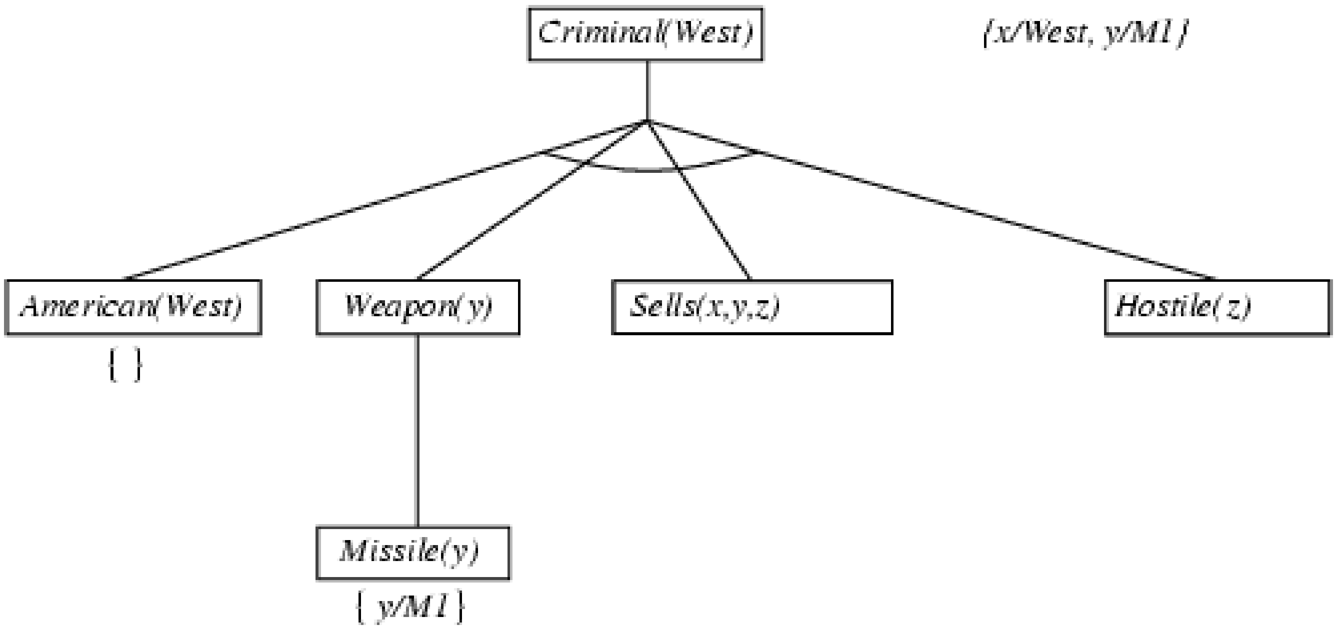
Backward chaining example



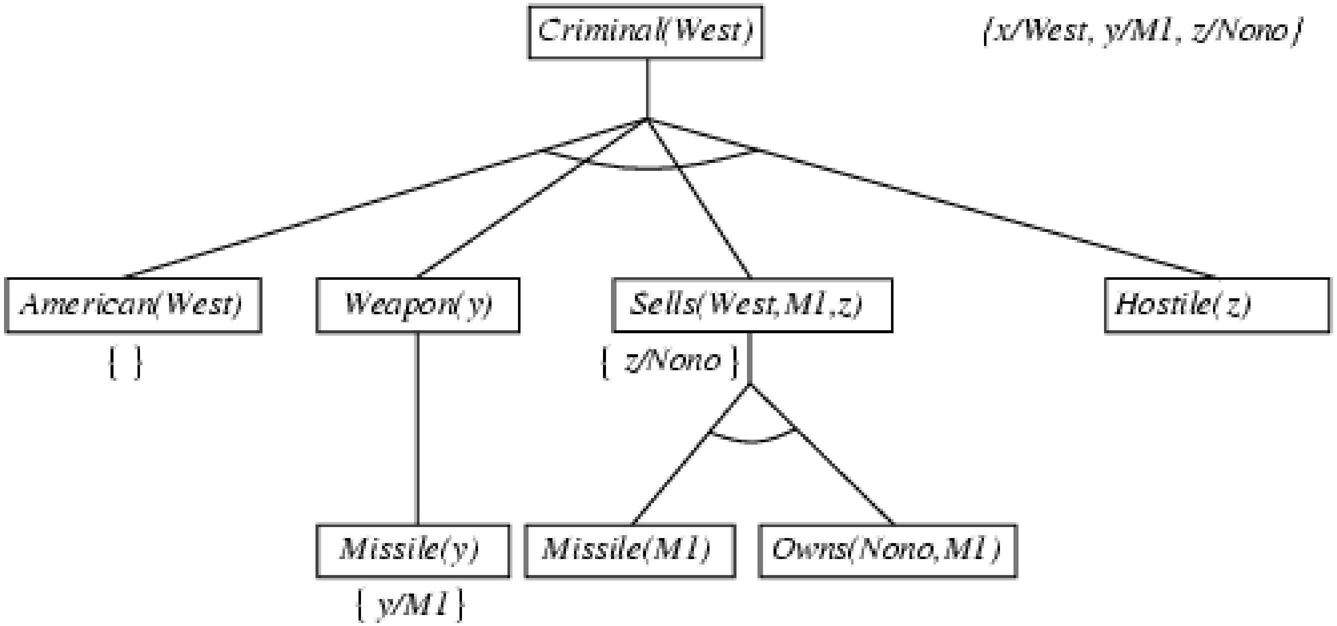
Backward chaining example



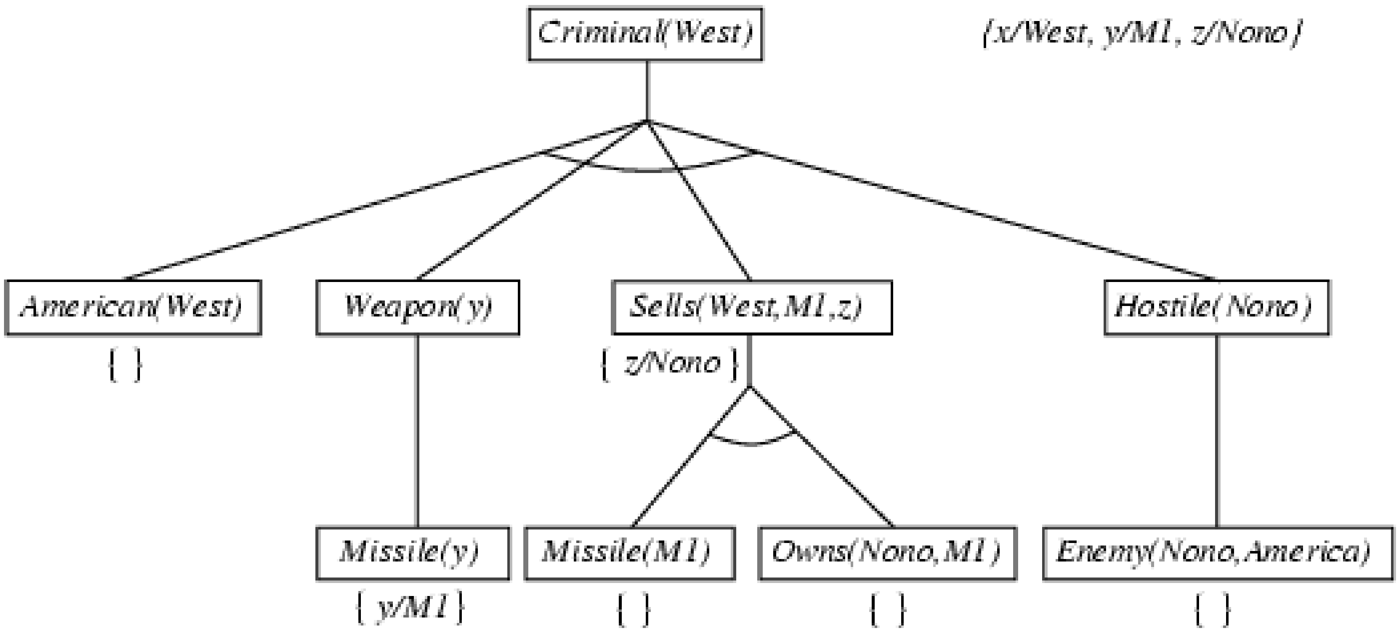
Backward chaining example



Backward chaining example



Backward chaining example



Properties of backward chaining

- Depth-first recursive proof search:
  - Space is linear in size of proof.
- Incomplete due to infinite loops
  - $\Rightarrow$ fix by checking current goal against every goal on stack
- Inefficient due to repeated subgoals (both success and failure)
  - $\Rightarrow$ fix using caching of previous results (memoization)
- Widely used for logic programming
- PROLOG:
  backward chaining with Horn clauses + bells & whistles.

Resolution in FOL

- Full first-order version:

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

where `Unify`(ℓ_i, $\neg m_j$) = θ.

- 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 θ = {x/Ken}

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

Converting FOL sentences to CNF

Original sentence:

Everyone who loves all animals is loved by someone:

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

1. Eliminate biconditionals and implications

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

2. Move $\neg$ inwards:

Recall: $\neg\forall x\ p \equiv \exists x\ \neg p$, $\neg\exists x\ p \equiv \forall x\ \neg p$

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

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

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

Conversion to CNF contd.

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

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

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

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

(reason: animal y could be a different animal for each x.)

Conversion to CNF contd.

5. Drop universal quantifiers:

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

   (all remaining variables assumed to be universally quantified)

6. Distribute $\vee$ over $\wedge$:

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

Original sentence is now in CNF form – can apply same ideas to all sentences in KB to convert into CNF

Also need to include negated query

Then use resolution to attempt to derive the empty clause which show that the query is entailed by the KB

Recall: Example Knowledge Base in FOL

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

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

... 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 ...

$American(West)$

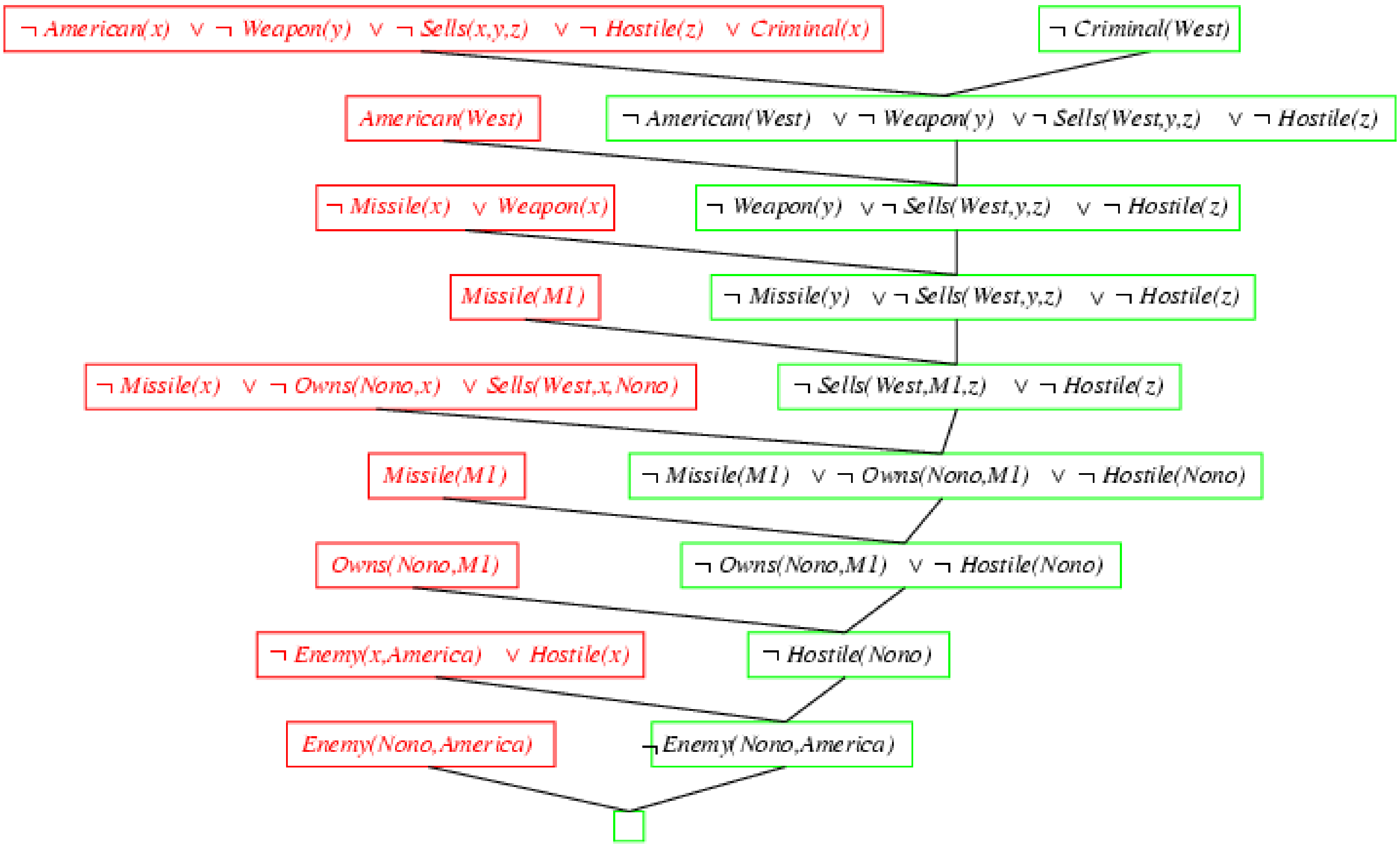
The country Nono, an enemy of America ...

$Enemy(Nono,America)$

Convert to CNF

Q: Criminal(West)?

Resolution proof



Second Example

KB:

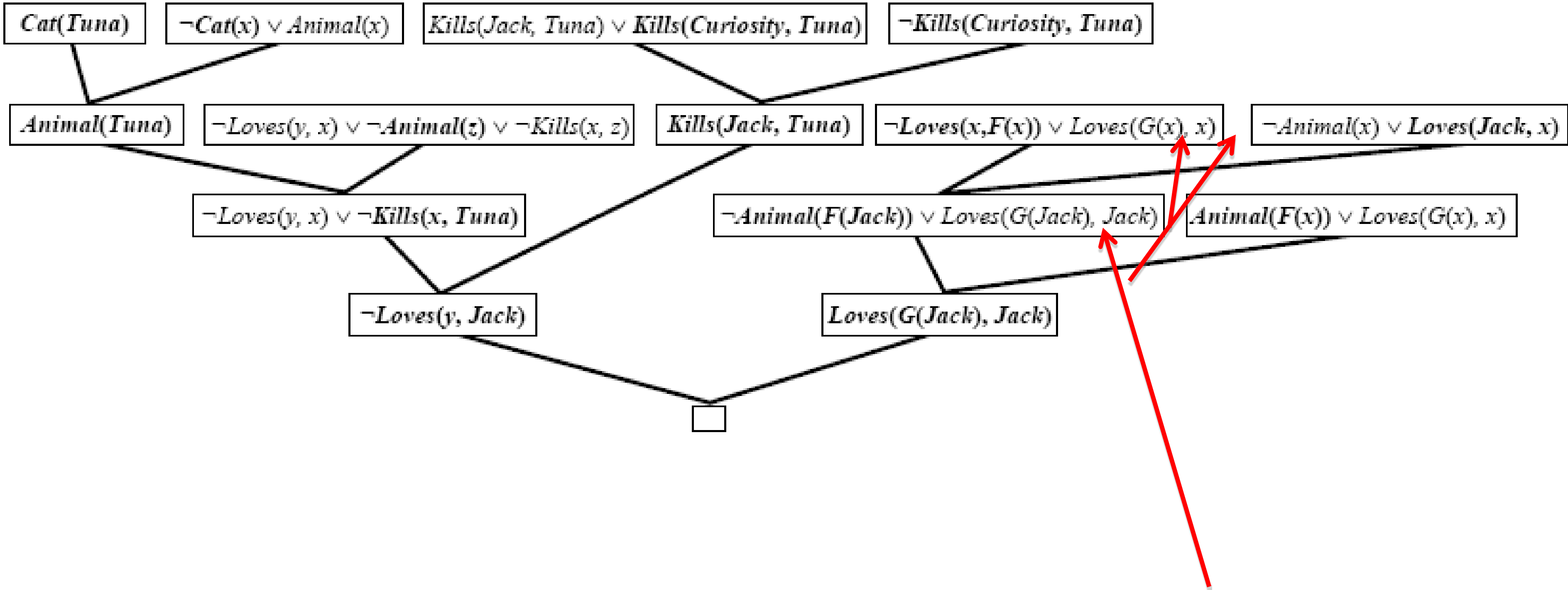
Everyone who loves all animals is loved by someone
Anyone who kills animals is loved by no-one
Jack loves all animals
Either Curiosity or Jack killed the cat, who is named Tuna

Query: *Did Curiousity kill the cat?*

Inference Procedure:

Express sentences in FOL
Convert to CNF form and negated query

Resolution-based Inference



Confusing because the sentences
Have not been standardized apart…

Other Types of Reasoning (all unsound, often useful)

- Inductive Reasoning (Induction)
  - Reason from a set of examples to the general principle.
  - Fact: You`ve liked all movies starring Meryl Streep.
    Inference: You'll like her next movie.
  - Basis for most learning and scientific reasoning.

- Abductive Reasoning (Abduction)
  - Reason from facts to the conclusion that best explains them.
  - Fact: A large amount of black smoke is coming from a home.
    Abduction1: The house is on fire.
    Abduction2: Bad cook.
  - Basis for most debugging and medical diagnosis.

- Analogical Reasoning (Analogy)
  - Reason from known (source) to unknown (target).
  - Fact: Water flow in a hose; pressure, constrictions.
    Inference: Electricity flow in a circuit; voltage, resistance.
  - Basis for much teaching.

Modal Logic Examples

- $\Box$ represents *Necessary*
  - Analogous to *“For All”*
- $\Diamond$ represents *Possible*
  - Analogous to *“There Exists”*

- $\Box \Leftrightarrow \neg\Diamond\neg$
- $\Diamond \Leftrightarrow \neg\Box\neg$ (Analogous to DeMorgan’s Law for Quantifiers)
- “It is *possible* that it will rain today.” $\Diamond$ RainToday
  “It is *not necessary* that it will *not* rain today.” $\neg\Box\neg$ RainToday

- Modal Logic of Knowledge and Belief.
  - $\Box$ represents “x *knows* that …”
  - $\Diamond$ represents “for all x knows, it may be true that …”
    - Equivalently, “x does *not know* that it is *not* true that …”
  - For reasoning about what other agents know and believe.
- Temporal Modal Logic
  - Modal operators [F] and [P] represent "henceforth" and "hitherto".
  - For reasoning about what will be and what has been.

Summary

- Inference in FOL
  - Simple approach: reduce all sentences to PL and apply propositional inference techniques
  - Generally inefficient

- FOL inference techniques
  - Unification
  - Generalized Modus Ponens
    - Forward-chaining
    - Backward-chaining
  - Resolution-based inference
    - Refutation-complete