First-Order Logic and Inference

Berlin Chen
Department of Computer Science & Information Engineering
National Taiwan Normal University

References:

1. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, Chapters 7,8 and 9
2. S. Russell's teaching materials

Pros and Cons of Propositional Logic (PL)

- PL is declarative
  - Pieces of syntax correspond to facts
  - Knowledge and inference are separate and inference is entirely domain-independent
- PL is compositional
  - Meaning of a sentence is a function of the meaning of its parts
  - E.g., meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$
- PL can deal with partial information
- The meaning of PL is context-independent
- PL has very limited expressive power
  - E.g., cannot say "pits cause breezes in adjacent squares" except by writing one sentence for each square

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$,
$B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$, ….

The lack of concise representations

Natural Languages

- Natural Languages are
  - Very expressive
  - Mediums for communication rather than pure representation
  - Context-dependent
  - Not purely compositional (e.g., “和尚” in Chinese)
  - Ambiguous (e.g., 三人參加 in Chinese=> 三-人參-加 or 三-人-參加)

- Major elements of natural Languages
  - Nouns and noun phrases: refer to objects
    - E.g., people, houses, colors, …
  - Verbs and verb phrases: refer to relations among objects
    - E.g., is red, is round, (properties)…; is brother of, has color, …
  - Some of the relations are functions which return one value for a given input
    - E.g., father of, best friend, beginning of, …

First-Order Logic (FOL)

- Whereas PL assumes world containing facts, FOL assumes the world contains objects and relations
  - FOL can express facts about some or all the objects in the universe
  - Such as “Squares neighboring the wumpus are smelly”

- Objects: things with individual identities

- Relations
  - Relations
  - Functions
    Interrelations among objects
  - Properties: distinguish objects from others

Logics in General

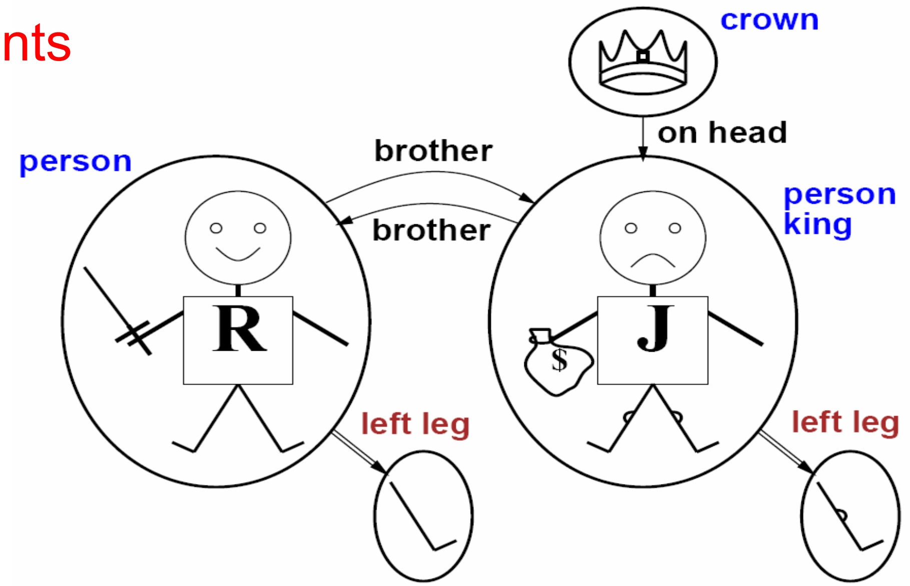
Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0,1]$
Fuzzy logic	degree of truth $\in [0,1]$	known interval value

Examples

- One plus two equals three
  - Objects: one, two, three, one plus two
  - Relation: equals
  - Function: plus
- Squares neighboring the wumpus are smelly
  - Objects: wumpus, square
  - Property: smelly
  - Relation: neighboring
- Evil King John ruled England in 1200
  - Objects: John, England, 1200
  - Properties: evil, king
  - Relation: ruled

Models for FOL

- A model contains objects and relations among them
  - PL: models are sets of truth values for proposition symbols
- The domain of a model is the set of objects
  - Objects are domain elements
- Example



5 objects: Richard, John, Richard's left legs, John's left legs, crown
2 binary relations: brother, on head
3 unary relations: person, king, crown
1 unary function: left leg

Syntax of FOL

- **BNF** (Backus-Naur Form) grammar for FOL

$$
\begin{aligned}
\textit{Sentence} \rightarrow\ & \textit{AtomicSentence} \\
& \mid (\textit{Sentence Connective Sentence}) \\
& \mid \textit{Quantifier Variables},\ \textit{Sentence} \\
& \mid \neg \textit{Sentence}
\end{aligned}
$$

$$\textit{AtomicSentence} \rightarrow \textit{Predicate}(\textit{Term}, \ldots) \mid \textit{Term} = \textit{Term}$$

relations, properties

$$
\begin{aligned}
\textit{Term} \rightarrow\ & \textit{Function}(\textit{Term}, \ldots) \\
& \mid \textit{Constant} \\
& \mid \textit{Variable}
\end{aligned}
$$

complex terms

$$
\begin{aligned}
\textit{Connective} &\rightarrow\ \Rightarrow \mid \wedge \mid \vee \mid \Leftrightarrow \\
\textit{Quantifier} &\rightarrow\ \forall \mid \exists \\
\textit{Constant} &\rightarrow\ A \mid X_1 \mid \textit{John} \mid \ldots \\
\textit{Variable} &\rightarrow\ a \mid x \mid s \mid \ldots \\
\textit{Predicate} &\rightarrow\ \textit{Before} \mid \textit{HasColor} \mid \textit{Raining} \mid \ldots \\
\textit{Function} &\rightarrow\ \textit{Mother} \mid \textit{LeftLeg} \mid \ldots
\end{aligned}
$$

Semantics of FOL

- The truth of any sentence is determined by a model and an interpretation for the sentence's symbols

- Interpretation specifies exactly which objects, relations and functions are referred by the constant, predicate, and function symbols
  - Constant symbols → objects
  - Predicate symbols → relations, properties
  - Function symbols → functional relations

- An atomic sentence *Predicate*($Term_1$, …,$Term_n$) is true iff the objects referred to by $Term_1$, …,$Term_n$ are in the relation referred to by *Predicate*

Terms

- A term is a logic expression that refers to an object
- Simple term: e.g., constant/variable symbols
- Complex term: formed by a function symbol followed a parenthesized list of terms as arguments to the function symbol
  - The complex term refers to an object that is the value of the function (symbol) applied to the arguments
  - E.g., *LeftLeg*(*John*)

function symbol argument/term

Atomic Sentences

- An atomic sentence is formed by
  - A predicate symbol followed by a parenthesized list of terms
    - *Predicate*(*Term*$_1$,…, *term*$_n$)
    - E.g., *Brother*(*Richard, John*)
  - Or just $term_1=term_2$
- Atomic sentences can have complex terms as arguments
  - E.g., *Married*(*Father*(*Richard*), *Mother*(*John*))

Complex Sentences

- An complex sentence is constructed using logical connectives
  - Negation

    $\neg Brother(LeftLeg(Richard), John)$
  - Conjunction

    $Brother(Richard, John) \wedge Brother(John, Richard)$
  - Disjunction

    $King(Richard) \vee King(John)$
  - Implication

    $\neg King(Richard) \Rightarrow King(John)$

The truth or falsehood of a complex sentence can be determined from the truth or falsehood of its component sentences

Universal Quantification

- The following sentence remains truth for all values of the variable

  $\forall \langle variable \rangle \; \langle sentence \rangle$

  – Variables are lowercase

  – E.g., "Everyone in Taiwan is industrious"

  $\forall x \; In(x, Taiwan) \Rightarrow Industrious(x)$

$\forall x \; P$ is true in a model m iff P with x being each possible object in the model
- Equivalent to the conjunction of instantiations of P

$$\begin{aligned} & In(Thomas, Taiwan) \Rightarrow Industrious(Thomas) \\ \wedge\, & In(Rich, Taiwan) \Rightarrow Industrious(Rich) \\ \wedge\, & In(Vicent, Taiwan) \Rightarrow Industrious(Vicent) \\ \wedge\, & In(Eileen, Taiwan) \Rightarrow Industrious(Eileen) \\ \wedge\, & \ldots\ldots \end{aligned}$$

Universal Quantification: A Common Mistake

- Typically, $\Rightarrow$ (implication) is the main connective with $\forall$

- Common mistake: using $\wedge$ as the main connective with $\forall$

  $\forall x \;\; In(x, Taiwan) \wedge Industrious(x)$

  Means " Everyone is in Taiwan and everyone is industrious"

an overly strong statement

Existential Quantification

- The following sentence remains true for some values of the variable

  $\exists \langle variable \rangle \ \langle sentence \rangle$

  - E.g., "Someone in Taiwan is industrious"

    $\exists x \ In(x, Taiwan) \wedge Industrious(x)$

$\exists x \ P$ is true in a model *m* iff *P* with *x* being each possible object in the model
- Equivalent to the disjunction of instantiations of *P*

$$
\begin{aligned}
& (In(Thomas, Taiwan) \wedge Industrious(Thomas)) \\
\vee\ & (In(Rich, Taiwan) \wedge Industrious(Rich)) \\
\vee\ & (In(Vicent, Taiwan) \wedge Industrious(Vicent)) \\
\vee\ & (In(Eileen, Taiwan) \wedge Industrious(Eileen)) \\
\vee\ & \ldots\ldots
\end{aligned}
$$

Existential Quantification : A Common Mistake

- Typically, $\wedge$ is the main connective with $\exists$

- Common mistake: using $\Rightarrow$ as the main connective with $\exists$

  $\exists x \; In(x, Taiwan) \Rightarrow Industrious(x)$

  Is true if there is anyone who is not in Taiwan

a overly weak statement

Properties of Quantifiers

- Nested Quantifiers (commutative!)

  $\forall x\ \forall y$ is the same as $\forall y\ \forall x$

  $\exists x\ \exists y$ is the same as $\exists y\ \exists x$

  $\exists x\ \forall y$ is the same as $\forall y\ \exists x$

  – Examples:
    - "There is a person who loves everyone in the world"

      $\exists x\ \forall y\ Loves(x, y)$
    - "Everyone in the world is loved by at least one person"

      $\forall y\ \exists x\ Loves(x, y)$

- Quantifier Duality
  – Each of the following sentences can be expressed using the other

  $\forall x\ Likes(x, IceCream) \Longleftrightarrow \neg \exists x\ \neg Likes(x, IceCream)$

  $\exists x\ Likes(x, IceCream) \Longleftrightarrow \neg \forall x\ \neg Likes(x, IceCream)$

Equality

- Make statements to the effect that two terms refer to the same object
  - Determine the truth of an equality sentence by seeing that the referents of the two terms are the same objects

  - E.g., state the facts about a given function

    $Father(John) = Henry$

  - E.g., insisting that two terms are not the same objects

    $\exists x \exists y\ Brother(x, Richard) \wedge Brother(y, Richard) \wedge \neg(x = y)$

    - Richard has at least two brothers

Review: De Morgan's Rules

$$\forall x \neg P \equiv \neg \exists x P$$

$$\neg \forall x P \equiv \exists x \neg P$$

$$\forall x P \equiv \neg \exists x \neg P$$

$$\exists x P \equiv \neg \forall x \neg P$$

$$\neg P \wedge \neg Q \equiv \neg (P \vee Q)$$

$$\neg (P \wedge Q) \equiv \neg P \vee \neg Q$$

$$P \wedge Q \equiv \neg (\neg P \vee \neg Q)$$

$$P \vee Q \equiv \neg (\neg P \wedge \neg Q)$$

Using First-Order Logic

- Assertions and Queries
  - Assertions:
    - Sentences are added to KB using TELL, such sentences are called *assertions*

      TELL(KB, $King(John)$)

      TELL(KB, $\forall x\ King(x) \Rightarrow Person(x)$)

  - Queries
    - Questions are asked using ASK, which are also called queries or goals

      ASK(KB, $King(John)$) $\Rightarrow$ return **true**

      ASK(KB, $Person(John)$) $\Rightarrow$ return **true**

      ASK(KB, $\exists x\ Person\ (x)$) $\Rightarrow$ return $\{x/John\}$

A substitution or binding list

Using First-Order Logic

- Example: The Kinship Domain
  - One's mother is one's female parent

    $\forall m,c\ Mother(m, c) \Leftrightarrow (Female(m) \wedge Parent(m, c))$
  - One's husband is one's male spouse

    $\forall w,h\ Husband(h, w) \Leftrightarrow (Male(h) \wedge Spouse(h, w))$
  - A grandparent is a a parent of one's parent

    $\forall g,c\ Grandparent(g, c) \Leftrightarrow (\exists p\ Parent(g, p) \wedge Parent(p, c))$
  - A sibling is another child of one's parents

    $\forall x,y\ Sibling(x, y) \Leftrightarrow x \neq y \wedge (\exists p\ Parent(p, x) \wedge Parent(p, y))$
  - A first cousin is a child of a parent's sibling

    $\forall x,y\ FirstCousin(x, y) \Leftrightarrow \exists p,ps\ Parent(p, x) \wedge Sibling(ps, p) \wedge Parent(ps, y)$

Interacting with FOL *KBs*

- Suppose a wumpus-world agent is using an FOL *KB* and perceives a stench and a breeze (but no glitter) at $t = 5$

stench breeze glitter bump scream

Tell(*KB, percept*([*Stench, Breeze, None, None, None*], 5))

Ask(*KB,* $\exists$ *a BestAction*(*a,* 5))

  - Does the KB entail any particular actions at $t = 5$?

    Answer: *Yes,* {*a/shoot*} A substitution or binding list

- Given a sentence *S* and a substitution θ, SUBST(θ, *S*) denotes the result of plugging θ into *S* ; e.g.,

S=*Smarter*(*x, y*)

θ={*x/Vicent*, *y/Thomas*}

SUBST(θ, S) =*Smarter*(*Vicent*, *Thomas*)

  - ASK(*KB, S)* returns some/all θ (substitutions or binding lists) such that *KB* |= SUBST(θ, S)

KB for the Wumpus World

- Perception

  stench breeze glitter bump scream

  $\forall t, s, g, m, c\ Percept([s, Breeze, g, m, c], t) \Rightarrow Breeze(t)$

  $\forall t, s, b, m, c\ Percept([s, b, Glitter, m, c], t) \Rightarrow Glitter(t)$

- Reflex

  $\forall t\ Glitter(t) \Rightarrow BestAction(Grab, t)$

- Environment

  $\forall x, y, a, b\ Adjacent([x,y], [a,b]) \Leftrightarrow$

  $[a,b] \in \{[x+1, y], [x-1, y], [x, y+1]\ [x, y-1]\}$

- Properties of agent's locations

  $\forall s, t\ At(Agent, s, t) \wedge Breeze(t) \Rightarrow Breezy(s)$

KB for the Wumpus World

- Square are breezy near a pit
  - **Diagnostic rule** – infer hidden causes from observable effects
    - If a square is breezy , some adjacent square must contain a pit

      $\forall s\ Breezy(s) \Rightarrow\ \exists\ r\ Adjacent(r, s) \wedge Pit(r)$
    - If a square is not breezy , no adjacent square contains a pit

      $\forall s\ \neg Breezy(s) \Rightarrow \neg\ \exists r\ Adjacent(r, s)\ \wedge Pit(r)$
    - Combined:

      $\forall s\ Breezy(s) \Leftrightarrow \exists\ r\ Adjacent(r, s) \wedge Pit(r)$
  - **Causal rule** – infer observable effects from hidden causes
    - A pit causes all adjacent squares to be breezy

      $\forall r\ Pit(r) \Rightarrow [\forall s\ Adjacent(r, s) \Rightarrow Breezy(s)\ ]$
    - If all squares adjacent to a given square are pitless, the square will not be breezy

      $\forall s\ [\forall r\ Adjacent(r, s) \Rightarrow \neg Pit(r)] \Rightarrow \neg\ Breezy(s)$
    - Combined:

      $\forall s\ Breezy(s) \Leftrightarrow \exists\ r\ Adjacent(r, s) \wedge Pit(r)$

model-based reasoning

Inference Rules for Quantifiers

- Substitution SUBST(θ , α)
  - Refer to applying the *substitution* θ to the sentence α
  - θ is a set of variable/(ground)term pairs

$$Person(x) \xrightarrow{\theta = \{x/John\}} Person(John)$$

SUBST(θ , α)

- Universal Instantiation (UI)
  - Infer any sentence obtained by substituting a ground term for the universally quantified variable
  - A ground term is a term without variable
    - could be a complex term

$$\frac{\forall v\ \alpha}{\text{SUBST}(\{v/g\},\ \alpha)} \xrightarrow{\theta = \{x/John\}} \frac{\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)}{King(John) \wedge Greedy(John) \Rightarrow Evil(John)}$$

Inference Rules for Quantifiers

- Existential Instantiation (EI)
  - Infer any sentence obtained by substituting a new constant symbol that does not appear elsewhere in the *KB* for the existentially quantified variable

$$\theta = \{x/C_1\}$$

$$\frac{\exists\ v\ \alpha}{\textsc{Subst}(\{v/k\},\ \alpha)} \quad \Longrightarrow \quad \frac{\exists x\ Crown(x) \wedge OnHead(x, John)}{Crown(C_1) \wedge OnHead(C1, John)}$$

  - A new constant symbol called Skolem constant

Universal/Existential Instantiation

- Universal instantiation can be applied several times to add new sentences
  - The new *KB* is logically equivalent to the old one

- Existential instantiation can be applied just once to replace the existential sentence
  - The new *KB* is not equivalent to the old one (?)
  - But is satisfiable iff the old *KB* was satisfiable

Reduction to Propositional Inference

- Suppose the KB contains:

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

- Instantiate the universal sentence in all possible ways:

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

- The new KB is propositionalized
  - View the ground atomic sentences as propositional symbols

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

Reduction to Propositional Inference

- Claims
  - A ground sentence is entailed by new *KB* iff entailed by original *KB*
  - Every FOL *KB* can be propositionalized so as to preserve entailment

- Idea
  - Propositionalize *KB* and query, apply resolution, return result

- Problem
  - When the *KB* includes a function symbol, there are infinitely many ground terms can be generated from substitutions
  - E.g., *Father*(*Father*(*Father*(*John*)))

Reduction to Propositional Inference

- Theorem: Herbrand (1930)
  - If a sentence is entailed by the original FOL *KB*, there is a proof involving just a finite subset of the propositionalized *KB*
- Idea:

  for n = 0 to ∞ do

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

  to see if the sentence α is entailed by this *KB*
- Problem
  - It works if α is entailed, and it loops if α is not entailed

Father(*John*) ⇨ *Father*(*Father*(*John*)) ⇨ *Father*(*Father*(*Father*(*John*))) ⇨ …….

depth 1 **depth 2** **depth 3** **depth *n***

Reduction to Propositional Inference

- Theorem: Turing (1936), Church (1936)
  - Entailment in FOL is semidecidable
    - Algorithms exists that say yes to every entailed sentence
      - The programs will halt
    - But no algorithm exists that also say no to every nonentailed sentence
      - The programs will stuck in a infinite loop
      - More deeply nested terms were generated

Problems with Propositionalization

- Propositionalization approach is rather inefficient
  - It seems to generate lots of irrelevant sentences
  - E.g., from

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

    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

Generalized Modus Ponens (GMP)

- For atomic sentences p_i, p_i' , and q , where there is a substitution θ such that SUBST(θ , p_i') = SUBST(θ , p_i) for all i

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

n atomic sentences p_i'
1 implication

atomic sentences

p_1' is $King(John)$

p_2' is $Greedy(y)$

p_1 is $King(x)$

p_2 is $Greedy(x)$

the premises of an implication

q is $Evil(x)$ $\longrightarrow$ $\text{SUBST}(\theta, q)$ is $Evil(John)$

θ is $\{x / John, y / John\}$: a set of variable/(ground)term pairs

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

  - GMP used with *KB* of definite clauses (exactly one positive literal)
  - All variables assumed universally quantified

Unification

- A process to find a substitution θ which can be applied to two sentences *p* and *q* to make them look the same

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

  – The UNIFY algorithm returns a unifier (θ) for the two sentences

- Example
  – Query: *Knows*(*John, x*)

  KB:
  Knows(*John, Jane*)
  Knows(*y, Bill*)
  Knows(*y, Mother*(*y*))
  Knows(*x, Elizabeth*)

p	***q***	θ

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*

Standardizing Apart

- Eliminate overlap of variables to avoid clashes by renaming variables

UNIFY(*Knows*(*John, x*), *Knows*(*x, Elizabeth*)) = *fail*

UNIFY(*Knows*(*John, x*), *Knows*(z_{17}, *Elizabeth*)) ={*x*/*Elizabeth*, z_{17}/*John*}

Most Generalized Unifier (MGU)

- Consider the following two unifications

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

  Knows(*John, z*)

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

  Knows(*John, John*)

  – We say the first unifier is more general than the second
    - It places fewer restrictions on the values of variables

- For every unifiable pairs of expressions, there is a single most generalized unifier (MGU)
  – E.g., the former unifier, {*y*/*John*, *x*/*z*}, shown above

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 (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(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

//Op: function symbol
// Args: argument list

---

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

As matching a variable against a complex term, check whether the variable itself occurs inside the term.
If it does, the match fails.

Efficient Indexing and Retrieval

- Predicate Indexing

Query: *Knows*(*John, x*) //an instance of "Fetching"

Employs(*x, Richard*)
Employs(*AIMA.org, y*)
Employs(*x, y*)

KB: *Knows*(*John, Helen*)
Brother(*John, Richard*)
Employs(*John*, *Richard*)
…..

  - Using a hash table
  - Maintain indices on keys composed of a predicate plus (one to several) arguments

Forward Chaining

- Operations
  - Start with the atomic sentences (known facts) in the *KB* and apply Generalized Modus Ponens in the forward direction (trigger rules whose premises are satisfied)
    - Unification of literals (Predicates in FOL)
  - Adding new atomic sentences (conclusions of implications)
    - Not just a renaming of a known fact
  - Repeat until the query is answered or no further inferences can be made
- To apply *FC*, the *KB* should be converted into a set of definite clauses

Definite Clauses

- Are disjunctions of literals, and of which exactly one is positive
  - A definite clause is a Horn clause with exact one positive literal

E.g., $\neg P1 \lor \neg P2 \lor \ldots \lor \neg Pn \lor Q$ can be converted to
$(P1 \land P2 \land \ldots \land Pn) \Rightarrow Q$

- More specifically, a definite clause is
  - Either an atomic clause (an positive literal)
  - Or an implication whose antecedent (premise/body) is a conjunction of positive literals and whose conclusion (head) is a single positive literal

$King(John)$
$Greedy(y)$
$King(x) \land Greedy(x) \Rightarrow Evil(x)$

  - Variables are assumed to be universally quantified

- Not all *KB* can be converted into a set of definite clauses
  - Because of the single-positive-literal restriction

Example *KB*

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

- Prove that West is a criminal

Criminal(*West*) *true* or *false* ?

Example *KB*

- It is a crime for an American to sell weapon to hostile nations

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

- The country Nono has some missiles

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

  $\Rightarrow Owns(Nono, M_1)$ 2 , $Missile(M_1)$ 3

  existential elimination/instantiation
  AND elimination

- All its (Nono's) missiles are sold to it by West

  $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$ 4

- Missiles are weapons

  $Missile(x) \Rightarrow Weapon(x)$ 5

- An enemy of America counts as "hostile"

  $Enemy(x, America) \Rightarrow Hostile(x)$ 6

A datalog KB:
composed of a set of
FOL definite clauses
with no function symbols

background knowledge!

Example *KB*

- West, who is American

  American(*West*) 7

- The country Nono, an enemy of America

  Enemy(*Nono*, *America*) 8

Example *KB*: FC Proof

- Start with the atomic sentences (known facts) in the *KB*

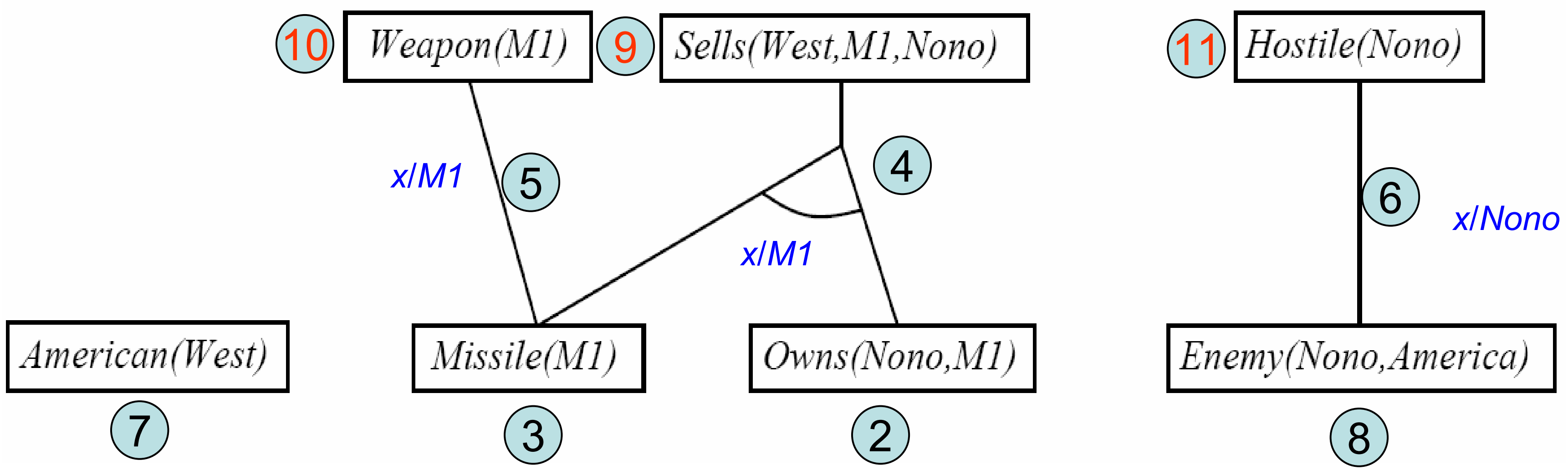
Proof Tree

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

Example *KB*: FC Proof

- Apply Generalized Modus Ponens in the forward direction to trigger rules whose premises are satisfied
- Adding new atomic sentences (conclusions)

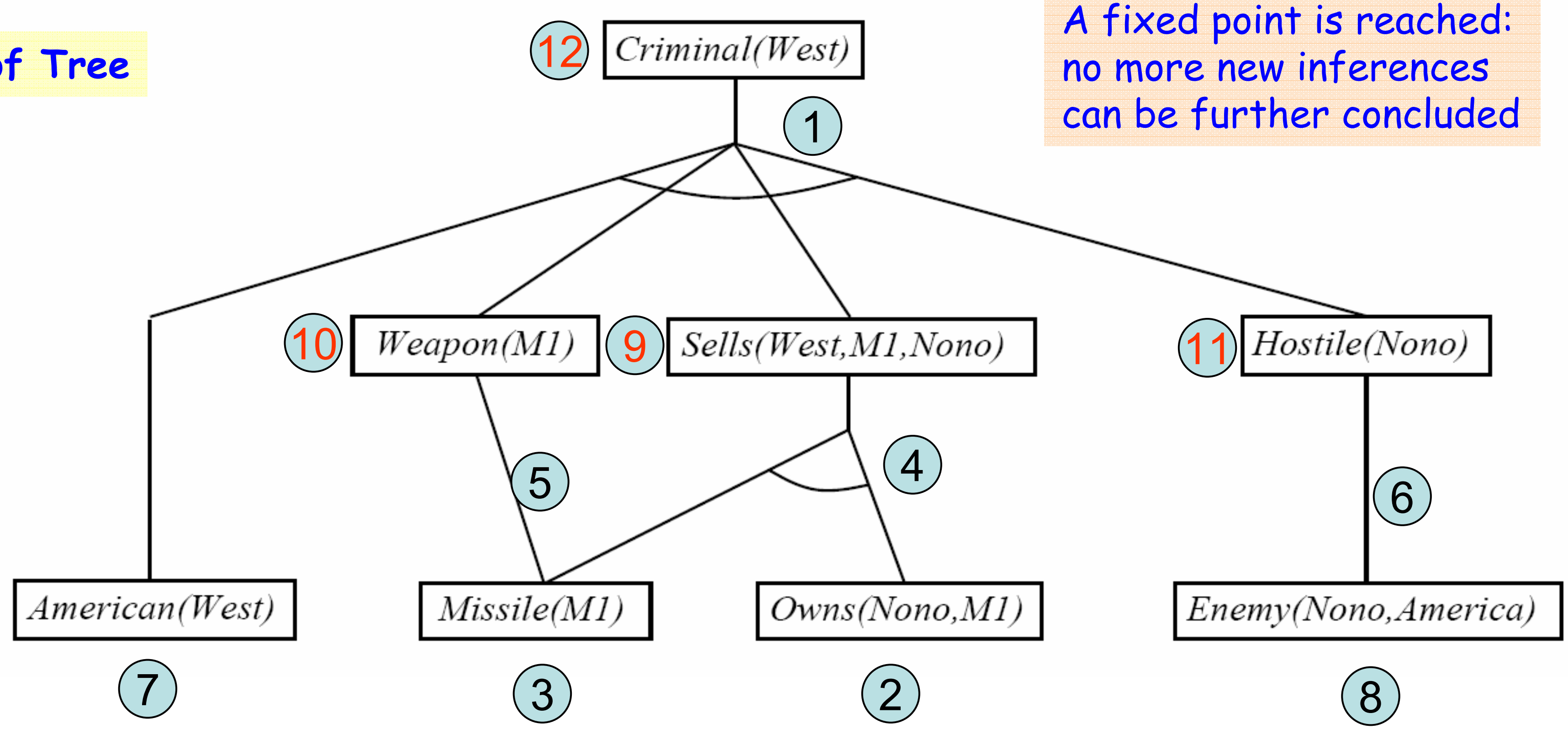
Proof Tree



Example *KB*: FC Proof

- Apply Generalized Modus Ponens in the forward direction to trigger rules whose premises are satisfied
- Adding new atomic sentences (conclusions)

Proof Tree



A fixed point is reached: no more new inferences can be further concluded

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
    $new \leftarrow \{\}$
    for each sentence r **in** KB **do** renaming the variables
      $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-APART(r) pattern matching
      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 atomic sentences
        $q' \leftarrow$ SUBST(θ, q)
        if q' is not a renaming of some sentence already in KB or new **then do**
          add q' to new
          $\boxed{\phi \leftarrow \text{UNIFY}(q', \alpha)}$ the new fact unified with the query
          if ϕ is not *fail* **then return** ϕ
    add new to KB
  return *false*

Forward Chaining Algorithm

- Problems
  - The inner loop (pattern matching) is very expensive
  - Rules will be rechecked on every iteration to see if its premises are satisfied
  - Many facts generated are irrelevant to the goal

Incremental Forward Chaining

- Every new fact inferred on iteration *t* must be derived from at least one new fact from iteration *t-1*
  - Check a rule only if its premise include a conjunct p_i can be unified with a fact p_i' newly inferred at iteration *t-1*
  - If so, fix p_i to match with p_i' and allow the other conjuncts of the rule to match with facts from any previous iteration

Properties of Forward Chaining

- FC is sound and complete for first-order definite clauses

- FC terminates for Datalog in poly iterations: (at most $p \cdot n^k$)
  - Datalog = first-order definite clauses + no functions

With p k-ary predicates and n constants

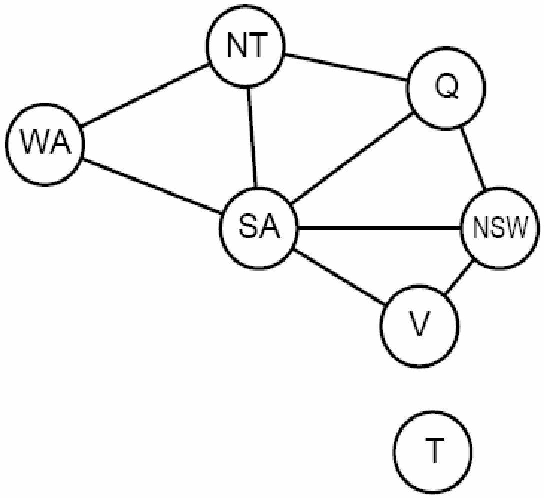
- May not terminate in general if α is not entailed
  - Entailment with datalog is decidable
  - Entailment with definite clauses is semi-decidable
    - When *KB* with functional symbols

$NatNum(0)$
$\forall n\ NatNum(n) \Rightarrow NatNum(S(n))$

Will add:
$NatNum(S(0))$, $NatNum(S(S(0)))$, $NatNum(S(S(S(0))))$, …

Hard Matching Example

- Express a finite-domain CSP as a single definite clause together with some associated ground facts
  - E.g., the map coloring problem



rule

$$
\begin{aligned}
&\mathit{Diff}(wa, nt) \wedge \mathit{Diff}(wa, sa) \wedge \\
&\quad \mathit{Diff}(nt, q) \wedge \mathit{Diff}(nt, sa) \wedge \\
&\quad \mathit{Diff}(q, nsw) \wedge \mathit{Diff}(q, sa) \wedge \\
&\quad \mathit{Diff}(nsw, v) \wedge \mathit{Diff}(nsw, sa) \wedge \\
&\quad \mathit{Diff}(v, sa) \;\Rightarrow\; \mathit{Colorable}()
\end{aligned}
$$

$$
\begin{array}{ll}
\mathit{Diff}(Red, Blue) & \mathit{Diff}(Red, Green) \\
\mathit{Diff}(Green, Red) & \mathit{Diff}(Green, Blue) \\
\mathit{Diff}(Blue, Red) & \mathit{Diff}(Blue, Green)
\end{array}
$$

Known facts

  - Matching a definite clause against a set of facts is *NP*-hard

Backward Chaining

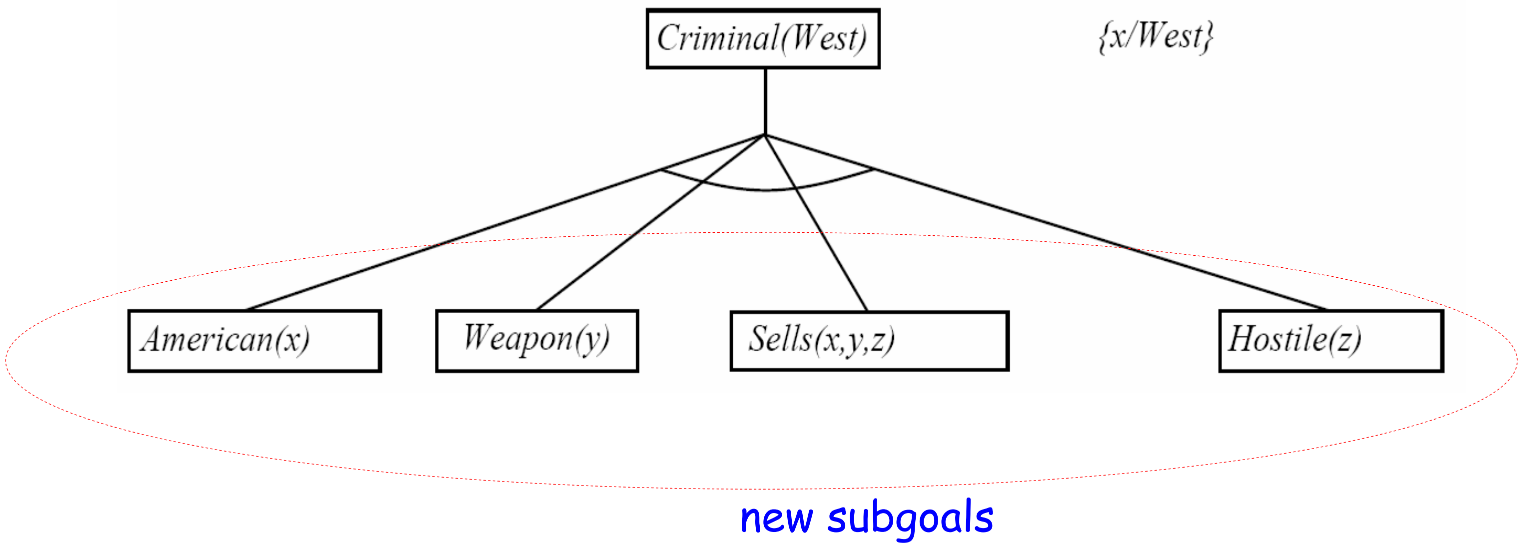
- Work backward from the goal (query), chaining through rules to find known facts that support the proof
  - Put the query on a stack
  - Pop it and find the set of all substitutions that satisfies the query
    - Find all implications in KB whose heads (conclusions) can be unified with the goals and put their bodies (premises) on the stack as new goals
    - Goals unified with known facts generate no new goals
  - If all goals on the stack are satisfied, (the current branch of) the proof succeeds

Example *KB*: BC Proof

$\boxed{Criminal(West)}$

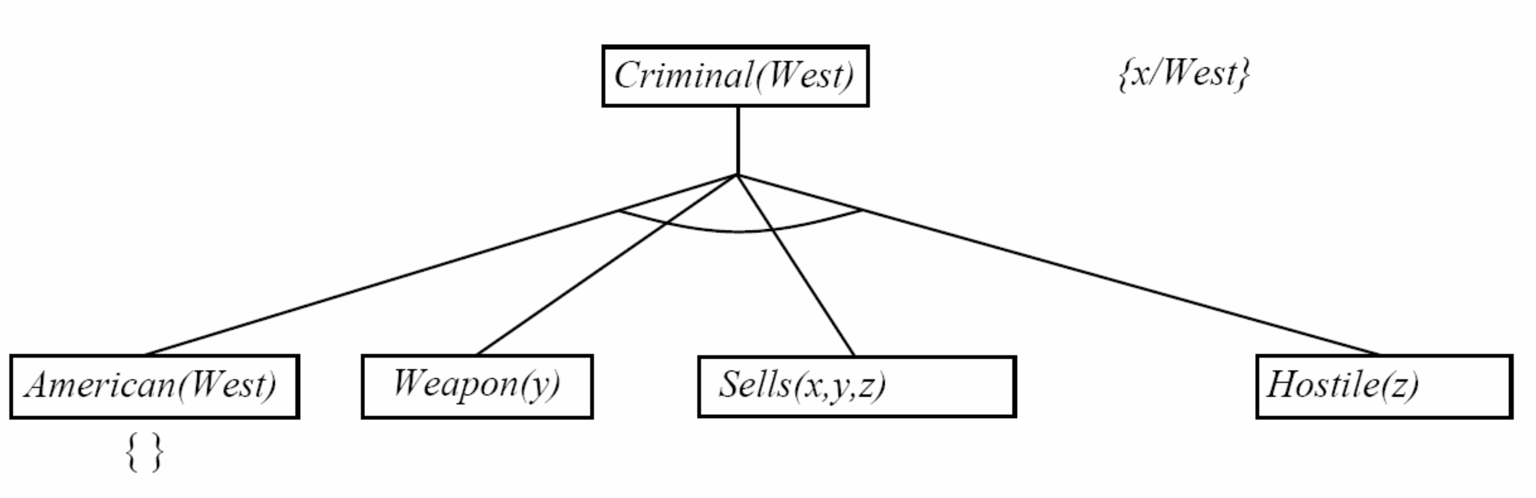
Put the query on a stack
Pup it and find the set of all substitutions that satisfies the query

Example *KB*: BC Proof



Proof Tree

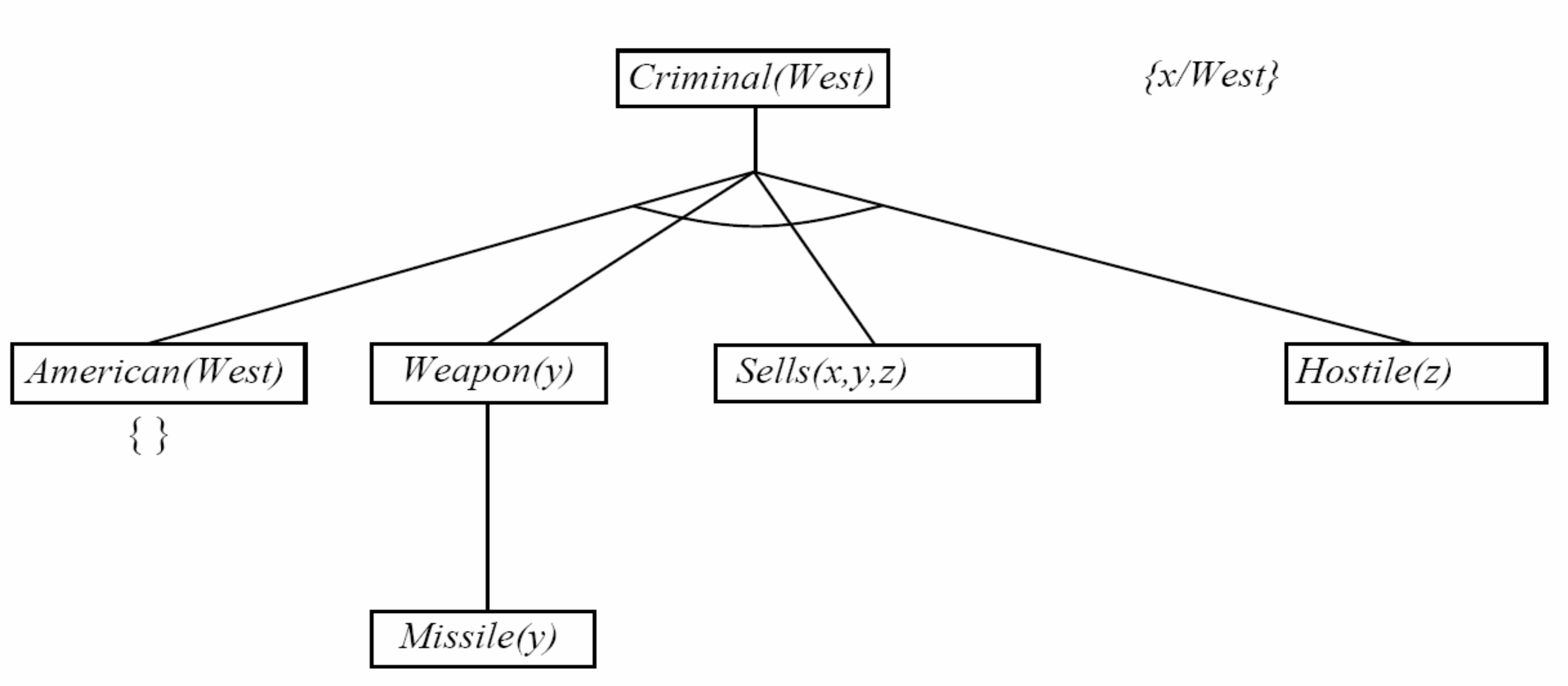
Example *KB*: BC Proof



Unified with
known fact

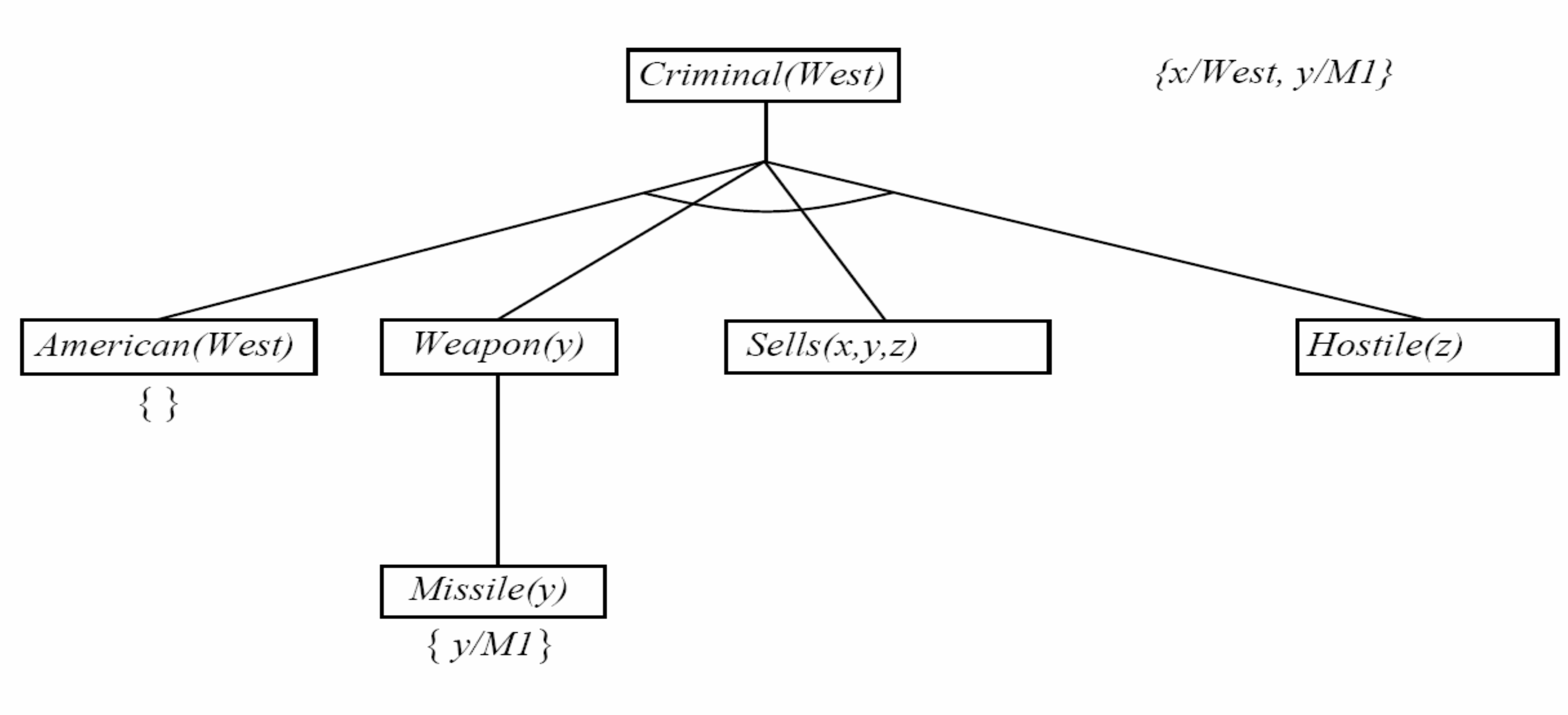
Proof Tree

Example *KB*: BC Proof



Proof Tree

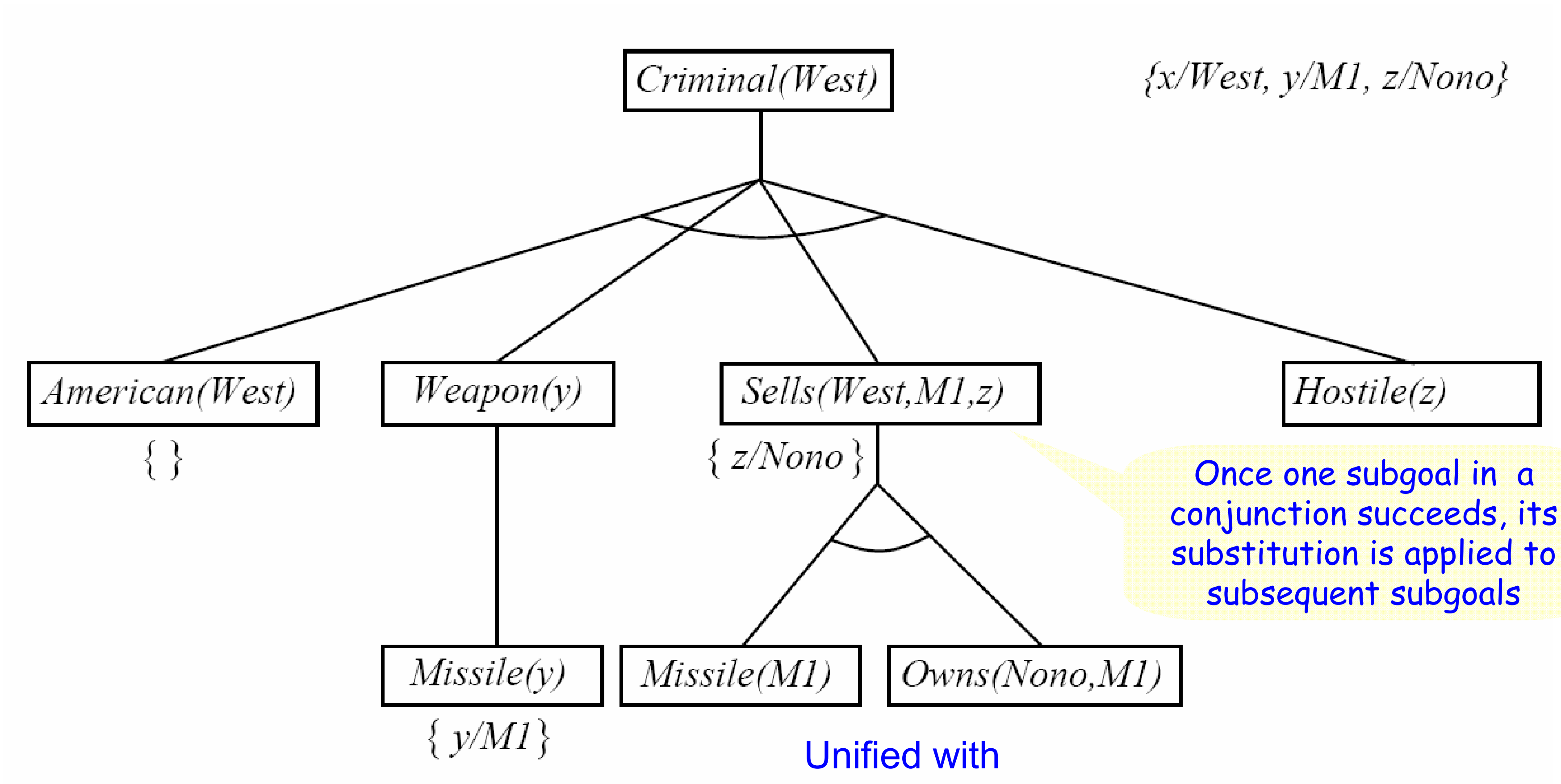
Example *KB*: BC Proof



Unified with
known fact

Proof Tree

Example *KB*: BC Proof



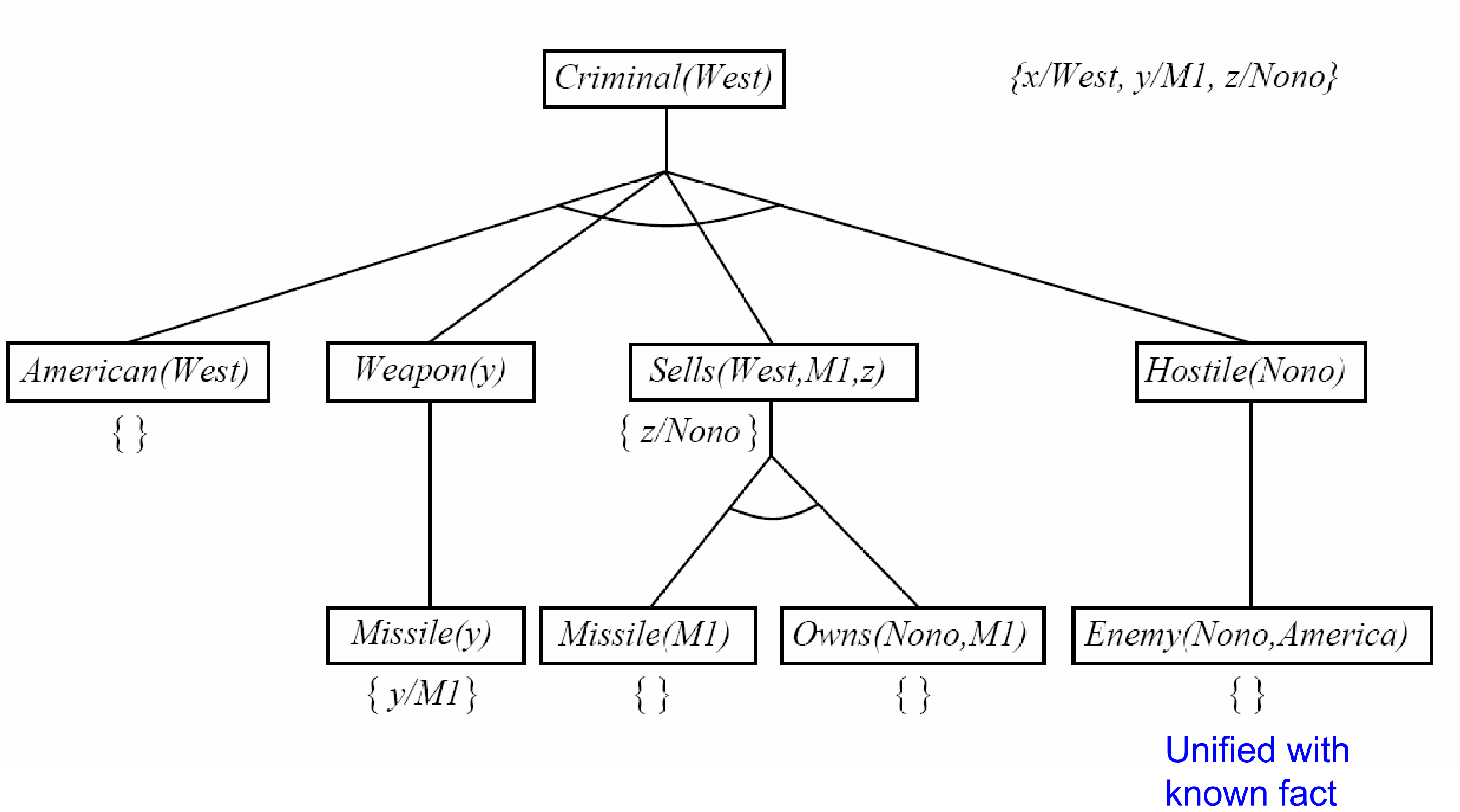
Once one subgoal in a conjunction succeeds, its substitution is applied to subsequent subgoals

Unified with known fact

$$\text{SUBST}(\text{COMPOSE}(\theta_1, \theta_2), p) = \text{SUBST}(\theta_2, \text{SUBST}(\theta_1, p))$$

Proof Tree

Example *KB*: BC Proof



Proof Tree

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** $\{\theta\}$
  $q' \leftarrow$ SUBST(θ, FIRST($goals$))
  for each sentence r **in** KB where STANDARDIZE-APART(r) = $(p_1 \wedge \ldots \wedge p_n \Rightarrow q)$
    and $\theta' \leftarrow$ UNIFY(q, q') succeeds
   $new\_goals \leftarrow [p_1, \ldots, p_n | \text{REST}(goals)]$ stack: LIFO
   $answers \leftarrow$ FOL-BC-ASK(KB, new_goals, COMPOSE(θ', θ)) $\cup$ $answers$
  return $answers$

Properties of Backward Chaining

- Depth-first recursive proof search
  - Space is linear in size of proof
- Incomplete due to infinite loops
  - Can be fixed by checking current goal against every goal on stack
- Inefficient due to repeated subgoals
  - Can be fixed by using caching of previous results (extra space !)

Conjunctive Normal Form (*CNF*) for *FOL*

- A *CNF* sentence in *FOL*
  - A conjunction (via $\wedge$'s operations) of clauses
  - Each clause is a disjunction (via $\vee$'s operations) of literals, where literals contain variables which are assumed to be universally quantified

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

CNF

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

- Every sentence of *FOL* can be converted into an inferentially equivalent *CNF*
  - The *CNF* sentence will be unsatisfiable if the oroginal one is unsatisfiable

Conversion to *CNF*

- Example :

  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 implications

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

- Move negation ($\neg$) inwards

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

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

- Standardize apart (renaming)

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

Conversion to *CNF*

- Skolemize (remove existential quantifier)

  $\forall x\, [Animal(A) \wedge \neg loves(x, A)\,] \vee loves(B, x)$ **?**

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

  – Existential variables replaced by skolem functions
  – The skolemized sentence is satisfiable when the original one is satisfiable

- Drop universal quantifiers

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

- Distribute conjunction($\wedge$) over disjunction ($\vee$)

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

Resolution

- The binary resolution rule for FOL can be express as

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

$$\text{where UNIFY}\left(l_i, \neg m_j\right) = \theta$$

Or

$$\frac{\begin{array}{c} l_1 \wedge \cdots \wedge l_k \Rightarrow p_1 \wedge \cdots \wedge p_s \\ q_1 \wedge \cdots \wedge q_r \Rightarrow m_1 \wedge \cdots \wedge m_n \end{array}}{\begin{array}{l} \mathrm{SUBST}(\theta,\ l_1 \wedge \cdots l_{i-1} \wedge l_{i+1} \cdots \wedge l_k \wedge q_1 \wedge \cdots \wedge q_r \\ \qquad \Rightarrow p_1 \wedge \cdots \wedge p_s \wedge m_1 \wedge \cdots m_{j-1} \wedge m_{j+1} \cdots \wedge m_n) \end{array}}$$

$$\text{where UNIFY}\left(l_i, m_j\right) = \theta$$

Resolution

- The combination of binary resolution rule and factoring is complete
  - Factoring: remove multiple copies of literals if they are unifiable (the unifier must be applied to the entire clause)

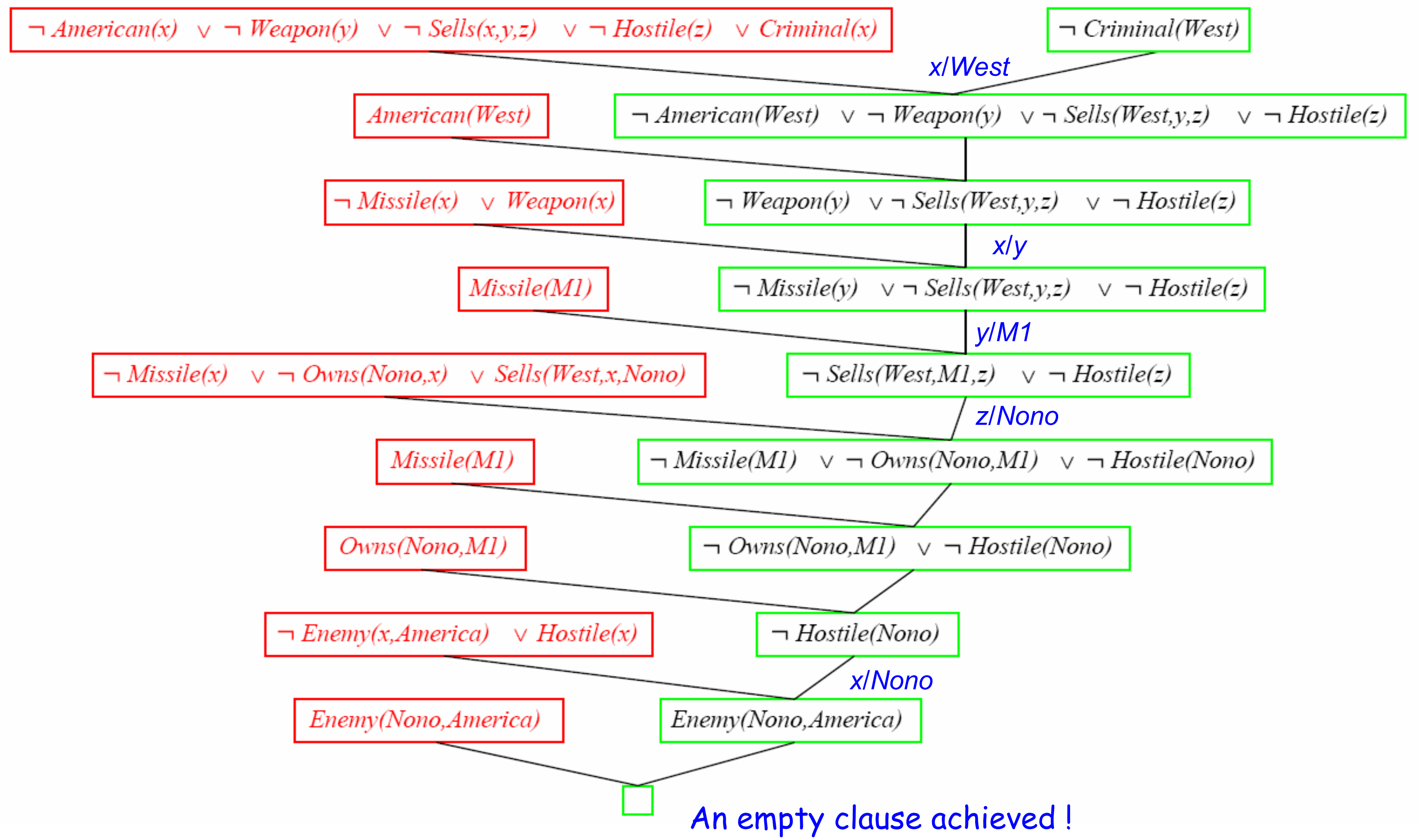
$[Animal(F(x)) \vee loves(G(x), x)]$ and $[\neg loves(u, v) \vee \neg Kills(u, v)]$

$\Theta = \{ u/G(x), v/x \}$

$\Rightarrow$ $[Animal(F(x)) \vee \neg Kills(G(x), x)]$

Resolution: Example Proof 1

- Proved by refutation



Resolution: Example Proof 2

Problem:

Everyone who loves all animals is loved by someone.

Anyone who kills an animal is loved by no one.

Jack loves all animals.

Either Jack or Curiosity killed the cat, who is named Tuna.

Did Curiosity kill the cat?

A. $\forall x\ [\forall y\ Animal(y) \Rightarrow loves(x, y)\ ] \Rightarrow\ \exists y\ loves(y, x)$
B. $\forall x\ [\forall y\ Animal(y) \Rightarrow Kills(x, y)\ ] \Rightarrow\ \forall z\ \neg loves(z, x)$
C. $\forall x\ Animal(x) \Rightarrow loves(Jack, x)$
D. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$
E. $Cat(Tuna)$
F. $\forall x\ Cat(x) \Rightarrow Animal(x)$ (background knowledge!)

$\neg$G. $\neg Kills(Curiosity, Tuna)$

Resolution: Example Proof 2

A1. $Animal(F(x)) \vee loves(G(x), x)$
A2. $\neg loves(x, F(x)) \vee loves(G(x), x)$
B. $\neg Animal(y) \vee \neg Kills(x, y) \vee \neg loves(z, x)$
C. $\neg Animal(x) \vee loves(Jack, x)$
D. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$
E. $Cat(Tuna)$
F. $\neg Cat(x) \vee Animal(x)$ (background knowledge!)

$\neg$G. $\neg Kills(Curiosity, Tuna)$

Resolution: Example Proof 2