# CECS545-Artificial Intelligence: First Order Logic

Dr. Roman Yampolskiy

# Outline

◇ Why FOL?

◇ Syntax and semantics of FOL

◇ Fun with sentences

◇ Wumpus world in FOL

# 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 |

# Pros and cons of propositional logic

- 😊 Propositional logic is **declarative**: pieces of syntax correspond to facts
- 😊 Propositional logic allows partial/disjunctive/negated information (unlike most data structures and databases)
- 😊 Propositional logic is **compositional**: meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$
- 😊 Meaning in propositional logic is **context-independent** (unlike natural language, where meaning depends on context)
- ☹ Propositional logic has very limited expressive power (unlike natural language)
  E.g., cannot say "pits cause breezes in adjacent squares"
  except by writing one sentence for each square

# First-order logic

Whereas propositional logic assumes world contains **facts**,
first-order logic (like natural language) assumes the world contains

- Objects: people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries . . .
- Relations: red, round, bogus, prime, multistoried . . ., brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, . . .
- Functions: father of, best friend, third inning of, one more than, end of . . .

# 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 |
| Fuzzy logic | facts + degree of truth | known interval value |

# Syntax of FOL: Basic elements

| | |
|---|---|
| Constants | $KingJohn, 2, UCB, \ldots$ |
| Predicates | $Brother, >, \ldots$ |
| Functions | $Sqrt, LeftLegOf, \ldots$ |
| Variables | $x, y, a, b, \ldots$ |
| Connectives | $\wedge \quad \vee \quad \neg \quad \Rightarrow \quad \Leftrightarrow$ |
| Equality | $=$ |
| Quantifiers | $\forall \; \exists$ |

# Atomic sentences

Atomic sentence $=$ $predicate(term_1, \ldots, term_n)$
or $term_1 = term_2$

Term $=$ $function(term_1, \ldots, term_n)$
or $constant$ or $variable$

E.g., $Brother(KingJohn, RichardTheLionheart)$
$> (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))$

# Complex sentences

Complex sentences are made from atomic sentences using connectives

$$\neg S, \quad S_1 \wedge S_2, \quad S_1 \vee S_2, \quad S_1 \;\Rightarrow\; S_2, \quad S_1 \;\Leftrightarrow\; S_2$$

E.g. $Sibling(KingJohn, Richard) \;\Rightarrow\; Sibling(Richard, KingJohn)$
$>(1,2) \vee \leq(1,2)$
$>(1,2) \wedge \neg{>}(1,2)$

# Truth in first-order logic

Sentences are true with respect to a model and an interpretation

Model contains $\geq 1$ objects (domain elements) and relations among them

Interpretation specifies referents for

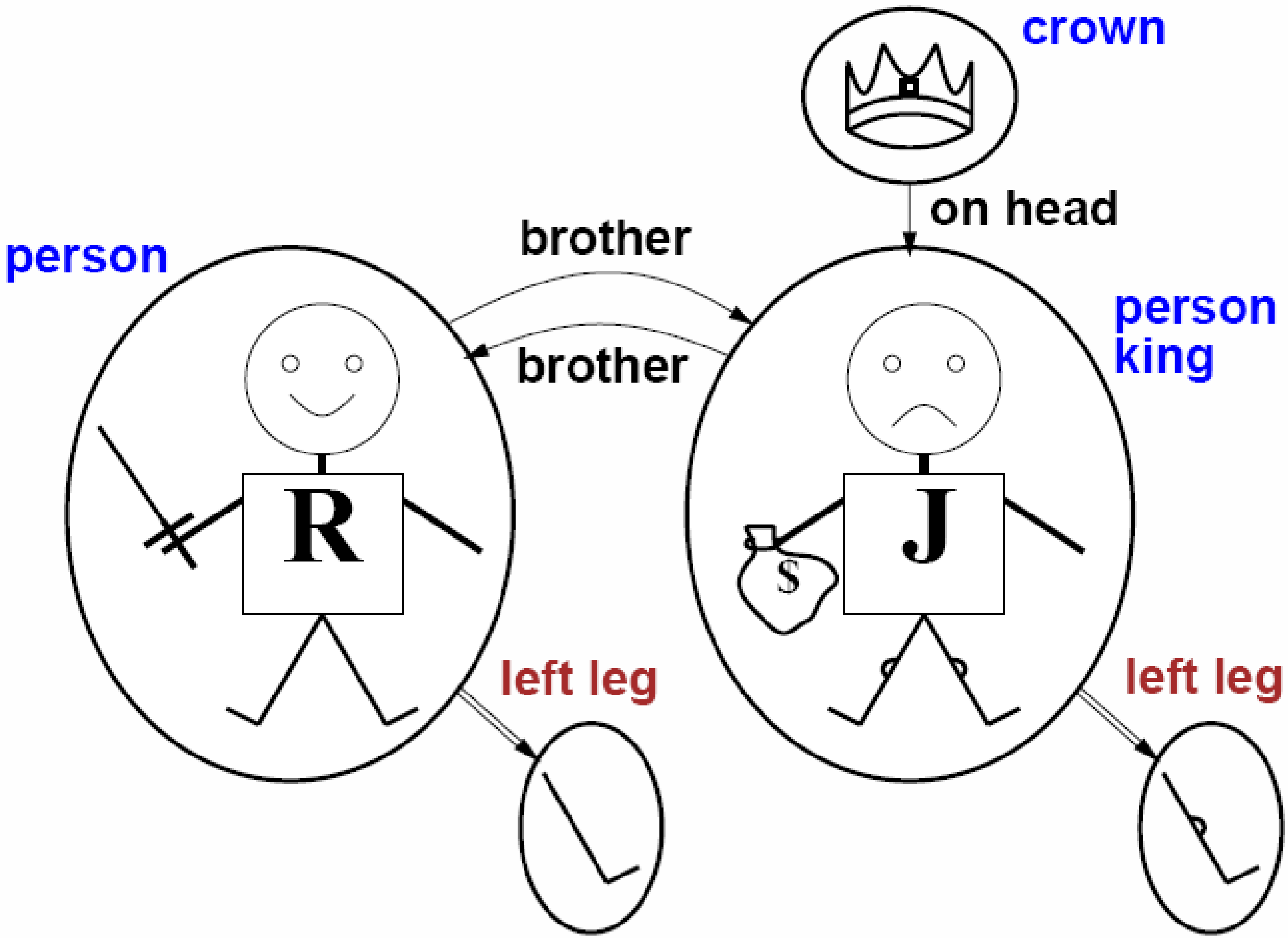
constant symbols $\rightarrow$ objects

predicate symbols $\rightarrow$ relations

function symbols $\rightarrow$ functional relations

An atomic sentence $predicate(term_1, \ldots, term_n)$ is true
iff the objects referred to by $term_1, \ldots, term_n$
are in the relation referred to by $predicate$

# Models for FOL: Example



# Truth example

Consider the interpretation in which

$Richard \rightarrow$ Richard the Lionheart

$John \rightarrow$ the evil King John

$Brother \rightarrow$ the brotherhood relation

Under this interpretation, $Brother(Richard, John)$ is true just in case Richard the Lionheart and the evil King John are in the brotherhood relation in the model

# Models for FOL: Lots!

Entailment in propositional logic can be computed by enumerating models

We **can** enumerate the FOL models for a given KB vocabulary:

For each number of domain elements $n$ from $1$ to $\infty$
    For each $k$-ary predicate $P_k$ in the vocabulary
        For each possible $k$-ary relation on $n$ objects
            For each constant symbol $C$ in the vocabulary
                For each choice of referent for $C$ from $n$ objects . . .

Computing entailment by enumerating FOL models is not easy!

# Universal quantification

$\forall \langle variables \rangle \ \langle sentence \rangle$

Everyone at Berkeley is smart:
$\forall x \ At(x, Berkeley) \Rightarrow Smart(x)$

$\forall x \ P$ is true in a model $m$ iff $P$ is true with $x$ being **each** possible object in the model

**Roughly** speaking, equivalent to the conjunction of instantiations of $P$

$$
\begin{aligned}
& (At(KingJohn, Berkeley) \Rightarrow Smart(KingJohn)) \\
\wedge\ & (At(Richard, Berkeley) \Rightarrow Smart(Richard)) \\
\wedge\ & (At(Berkeley, Berkeley) \Rightarrow Smart(Berkeley)) \\
\wedge\ & \ldots
\end{aligned}
$$

# Existential quantification

$\exists \langle variables \rangle \ \langle sentence \rangle$

Someone at Stanford is smart:
$\exists x \ At(x, Stanford) \land Smart(x)$

$\exists x \ P$ is true in a model $m$ iff $P$ is true with $x$ being **some** possible object in the model

**Roughly** speaking, equivalent to the disjunction of instantiations of $P$

$$
\begin{aligned}
& (At(KingJohn, Stanford) \land Smart(KingJohn)) \\
\lor \ & (At(Richard, Stanford) \land Smart(Richard)) \\
\lor \ & (At(Stanford, Stanford) \land Smart(Stanford)) \\
\lor \ & \ldots
\end{aligned}
$$

# Another common mistake to avoid

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

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

$$\exists\, x \;\; At(x, Stanford) \Rightarrow Smart(x)$$

is true if there is anyone who is not at Stanford!

# Properties of quantifiers

$\forall x\ \forall y$ 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 **not** the same as $\forall y\ \exists x$

$\exists x\ \forall y\ \ Loves(x, y)$
"There is a person who loves everyone in the world"

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

Quantifier duality: each can be expressed using the other

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

$\exists x\ \ Likes(x, Broccoli) \qquad \neg\forall x\ \ \neg Likes(x, Broccoli)$

# Fun with sentences

Brothers are siblings

$\forall x, y \;\; Brother(x, y) \;\Rightarrow\; Sibling(x, y)$.

"Sibling" is symmetric

$\forall x, y \;\; Sibling(x, y) \;\Leftrightarrow\; Sibling(y, x)$.

One's mother is one's female parent

$\forall x, y \;\; Mother(x, y) \;\Leftrightarrow\; (Female(x) \wedge Parent(x, y))$.

A first cousin is a child of a parent's sibling

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

# Equality

$term_1 = term_2$ is true under a given interpretation
if and only if $term_1$ and $term_2$ refer to the same object

E.g., $1 = 2$ and $\forall\, x \;\; \times(Sqrt(x), Sqrt(x)) = x$ are satisfiable
$2 = 2$ is valid

E.g., definition of (full) $Sibling$ in terms of $Parent$:

$$\forall\, x, y \;\; Sibling(x, y) \;\Leftrightarrow\; [\neg(x = y) \land \exists\, m, f \;\; \neg(m = f) \land$$
$$Parent(m, x) \land Parent(f, x) \land Parent(m, y) \land Parent(f, y)]$$

# Interacting with FOL KBs

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

$Tell(KB, Percept([Smell, Breeze, None], 5))$
$Ask(KB, \exists a \;\; Action(a, 5))$

I.e., does $KB$ entail any particular actions at $t = 5$?

Answer: $Yes, \;\; \{a/Shoot\}$ $\leftarrow$ substitution (binding list)

Given a sentence $S$ and a substitution $\sigma$,
$S\sigma$ denotes the result of plugging $\sigma$ into $S$; e.g.,
$S = Smarter(x, y)$
$\sigma = \{x/Hillary, y/Bill\}$
$S\sigma = Smarter(Hillary, Bill)$

$Ask(KB, S)$ returns some/all $\sigma$ such that $KB \models S\sigma$

# Knowledge base for the wumpus world

"Perception"

$\forall\, b, g, t\;\; Percept([Smell, b, g], t) \Rightarrow Smelt(t)$

$\forall\, s, b, t\;\; Percept([s, b, Glitter], t) \Rightarrow AtGold(t)$

Reflex: $\forall\, t\;\; AtGold(t) \Rightarrow Action(Grab, t)$

Reflex with internal state: do we have the gold already?

$\forall\, t\;\; AtGold(t) \wedge \neg Holding(Gold, t) \Rightarrow Action(Grab, t)$

$Holding(Gold, t)$ cannot be observed

$\Rightarrow$ keeping track of change is essential

# Deducing hidden properties

Properties of locations:

$\forall x, t \;\; At(Agent, x, t) \wedge Smelt(t) \;\Rightarrow\; Smelly(x)$

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

Squares are breezy near a pit:

Diagnostic rule—infer cause from effect

$$\forall y \;\; Breezy(y) \;\Rightarrow\; \exists x \;\; Pit(x) \wedge Adjacent(x, y)$$

Causal rule—infer effect from cause

$$\forall x, y \;\; Pit(x) \wedge Adjacent(x, y) \;\Rightarrow\; Breezy(y)$$

Neither of these is complete—e.g., the causal rule doesn't say whether squares far away from pits can be breezy

Definition for the $Breezy$ predicate:

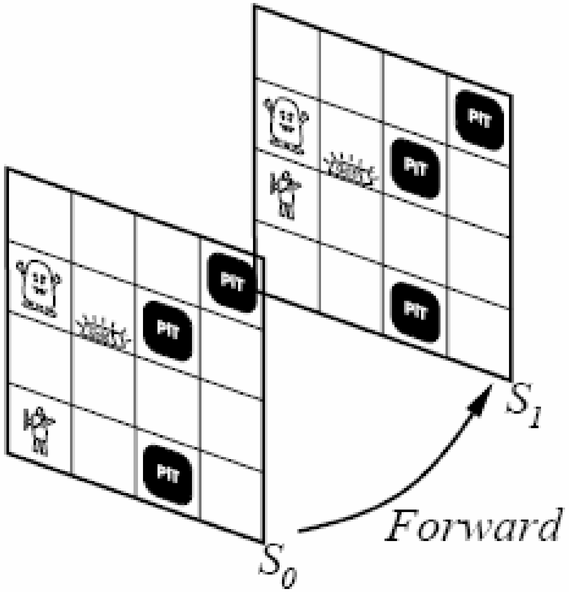
$$\forall y \;\; Breezy(y) \;\Leftrightarrow\; [\exists x \;\; Pit(x) \wedge Adjacent(x, y)]$$

# Keeping track of change

Facts hold in situations, rather than eternally
E.g., $Holding(Gold, Now)$ rather than just $Holding(Gold)$

Situation calculus is one way to represent change in FOL:
Adds a situation argument to each non-eternal predicate
E.g., $Now$ in $Holding(Gold, Now)$ denotes a situation

Situations are connected by the $Result$ function
$Result(a, s)$ is the situation that results from doing $a$ in $s$



# Describing actions I

"Effect" axiom—describe changes due to action
$\forall s \;\; AtGold(s) \;\Rightarrow\; Holding(Gold, Result(Grab, s))$

"Frame" axiom—describe **non-changes** due to action
$\forall s \;\; HaveArrow(s) \;\Rightarrow\; HaveArrow(Result(Grab, s))$

Frame problem: find an elegant way to handle non-change
- (a) representation—avoid frame axioms
- (b) inference—avoid repeated "copy-overs" to keep track of state

Qualification problem: true descriptions of real actions require endless caveats—what if gold is slippery or nailed down or . . .

Ramification problem: real actions have many secondary consequences—what about the dust on the gold, wear and tear on gloves, . . .

# Describing actions II

Successor-state axioms solve the representational frame problem

Each axiom is "about" a **predicate** (not an action per se):

$$\begin{array}{lll} \text{P true afterwards} & \Leftrightarrow & [\text{an action made P true} \\ & \vee & \text{P true already and no action made P false}] \end{array}$$

For holding the gold:

$$\begin{array}{l} \forall\, a, s\;\; Holding(Gold, Result(a, s)) \;\Leftrightarrow \\ \quad [(a = Grab \wedge AtGold(s)) \\ \quad \vee\, (Holding(Gold, s) \wedge a \neq Release)] \end{array}$$

# Making plans

Initial condition in KB:

$At(Agent, [1,1], S_0)$
$At(Gold, [1,2], S_0)$

Query: $Ask(KB, \exists s \ \ Holding(Gold, s))$
i.e., in what situation will I be holding the gold?

Answer: $\{s/Result(Grab, Result(Forward, S_0))\}$
i.e., go forward and then grab the gold

This assumes that the agent is interested in plans starting at $S_0$ and that $S_0$ is the only situation described in the KB

# Making plans: A better way

Represent plans as action sequences $[a_1, a_2, \ldots, a_n]$

$PlanResult(p, s)$ is the result of executing $p$ in $s$

Then the query $Ask(KB, \exists\, p\;\; Holding(Gold, PlanResult(p, S_0)))$
has the solution $\{p/[Forward, Grab]\}$

Definition of $PlanResult$ in terms of $Result$:

$$\forall\, s\;\; PlanResult([\,], s) = s$$

$$\forall\, a, p, s\;\; PlanResult([a|p], s) = PlanResult(p, Result(a, s))$$

Planning systems are special-purpose reasoners designed to do this type of inference more efficiently than a general-purpose reasoner

# Inference in first-order logic

Chapter 9

# Universal instantiation (UI)

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

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

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

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

$$King(John) \land Greedy(John) \;\Rightarrow\; Evil(John)$$
$$King(Richard) \land Greedy(Richard) \;\Rightarrow\; Evil(Richard)$$
$$King(Father(John)) \land Greedy(Father(John)) \;\Rightarrow\; Evil(Father(John))$$
$$\vdots$$

# Existential instantiation (EI)

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

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

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

Another example: from $\exists\, x \;\; d(x^y)/dy = x^y$ we obtain

$$d(e^y)/dy = e^y$$

provided $e$ is a new constant symbol

# Existential instantiation contd.

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

EI can be applied once to **replace** the existential sentence;
the new KB is **not** equivalent to the old,
but is satisfiable iff the old KB was satisfiable

# Reduction to propositional inference

Suppose the KB contains just the following:

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

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

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

The new KB is propositionalized: proposition symbols are

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

# Problems with propositionalization

Propositionalization 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

With function symbols, it gets nuch much worse!

# Unification

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

$\theta = \{x/John, y/John\}$ works

$\textsc{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, OJ)$ | $\{x/OJ, y/John\}$ |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | $\{y/John, x/Mother(John)\}$ |
| $Knows(John, x)$ | $Knows(x, OJ)$ | $fail$ |

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

# Generalized Modus Ponens (GMP)

$$\frac{p_1{}',\ \ p_2{}',\ \ldots,\ p_n{}',\ \ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta} \qquad \text{where } p_i{}'\theta = p_i\theta \text{ for all } i$$

| | |
|---|---|
| $p_1{}'$ is $King(John)$ | $p_1$ is $King(x)$ |
| $p_2{}'$ is $Greedy(y)$ | $p_2$ is $Greedy(x)$ |
| $\theta$ is $\{x/John, y/John\}$ | $q$ is $Evil(x)$ |
| $q\theta$ is $Evil(John)$ | |

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

All variables assumed universally quantified

# 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:

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

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

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

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

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

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

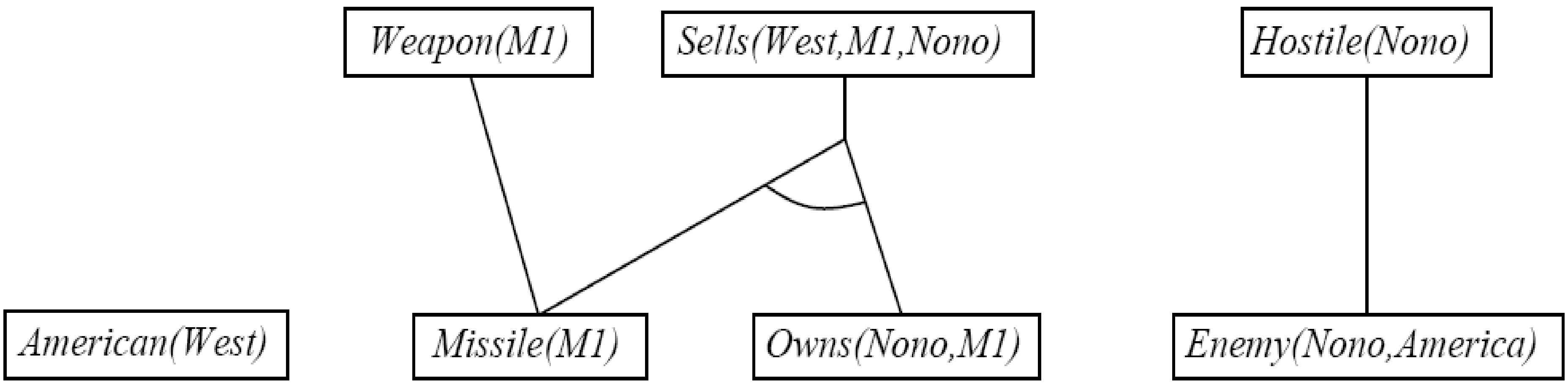
$American(West)$

$Missile(M1)$

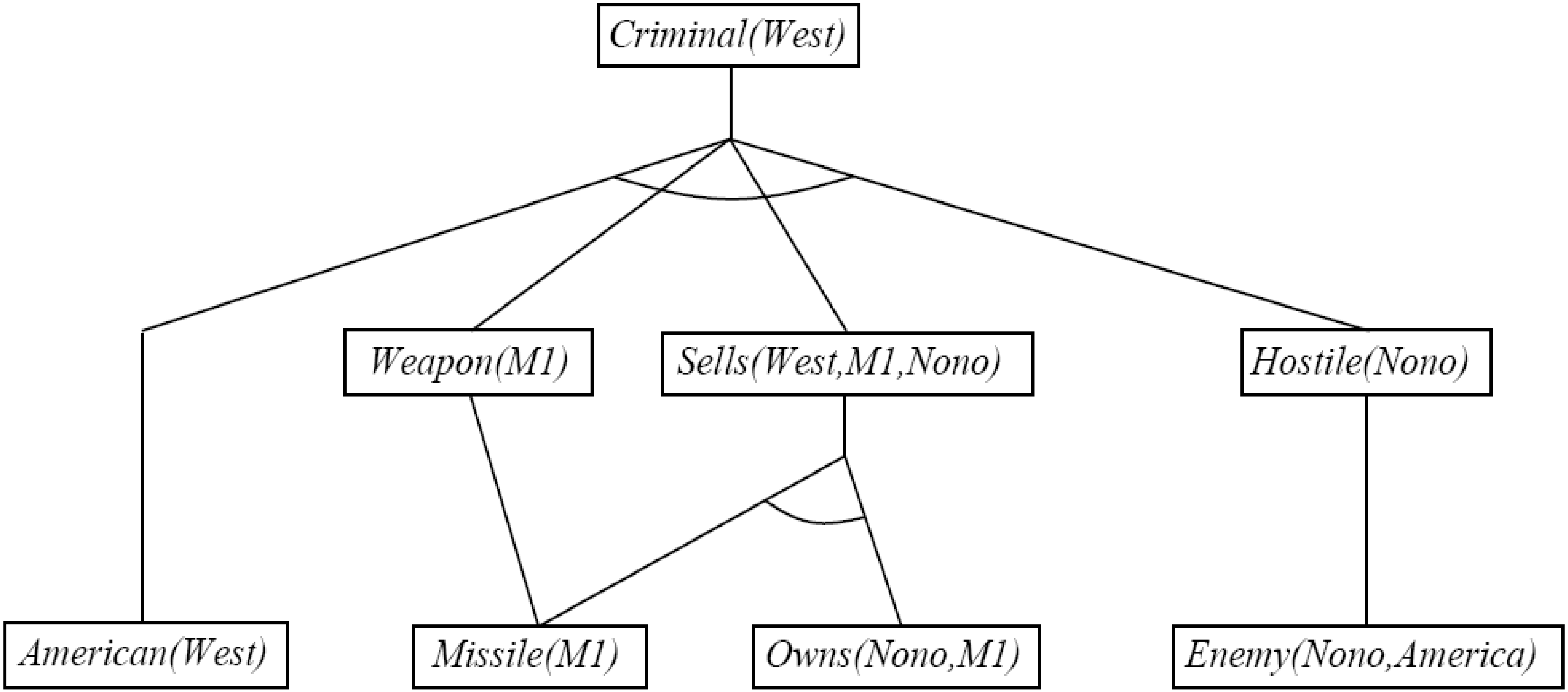
$Owns(Nono,M1)$

$Enemy(Nono,America)$

# Forward chaining proof



# Forward chaining proof



# Efficiency of forward chaining

Simple observation: 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 literal

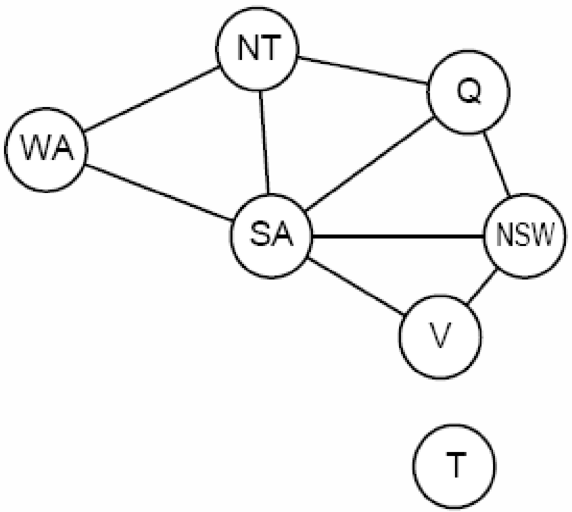
Matching itself can be expensive

Database indexing allows $O(1)$ retrieval of known facts
e.g., query $Missile(x)$ retrieves $Missile(M_1)$

Matching conjunctive premises against known facts is NP-hard

Forward chaining is widely used in deductive databases

# Hard matching example




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

$Diff(Red, Blue)$ $Diff(Red, Green)$
$Diff(Green, Red)$ $Diff(Green, Blue)$
$Diff(Blue, Red)$ $Diff(Blue, Green)$

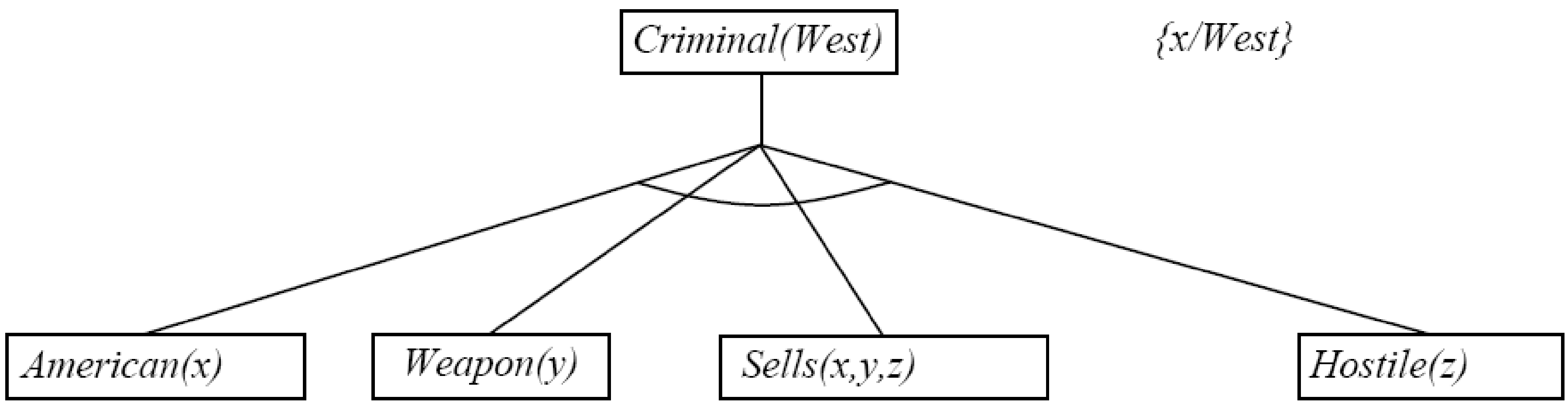
$Colorable()$ is inferred iff the CSP has a solution

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

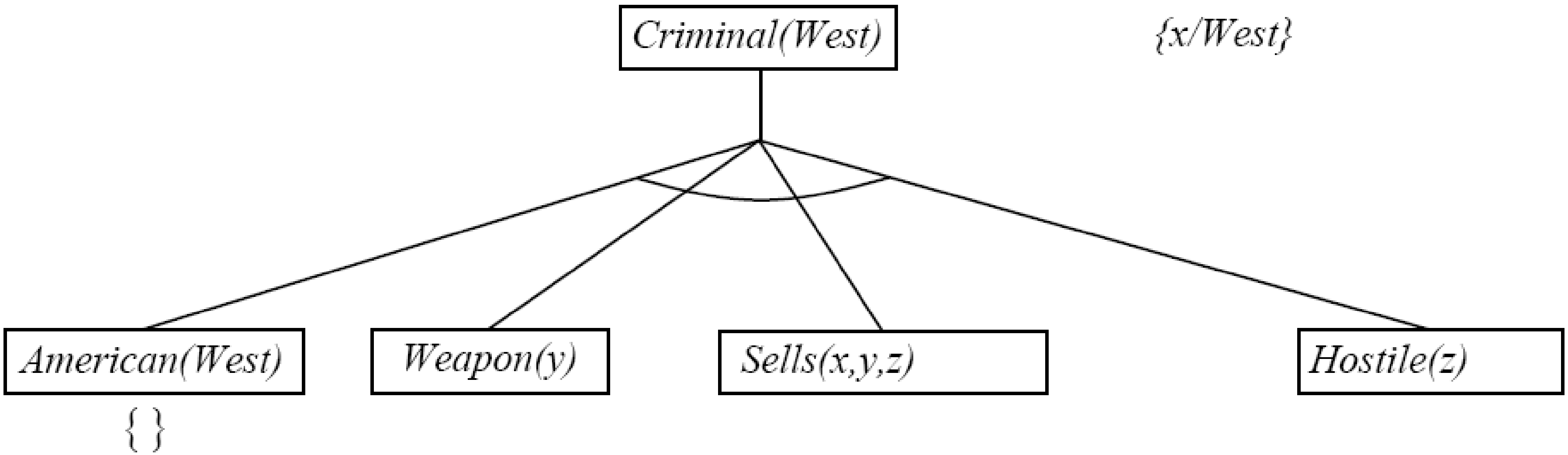
# Backward chaining example

$\boxed{Criminal(West)}$

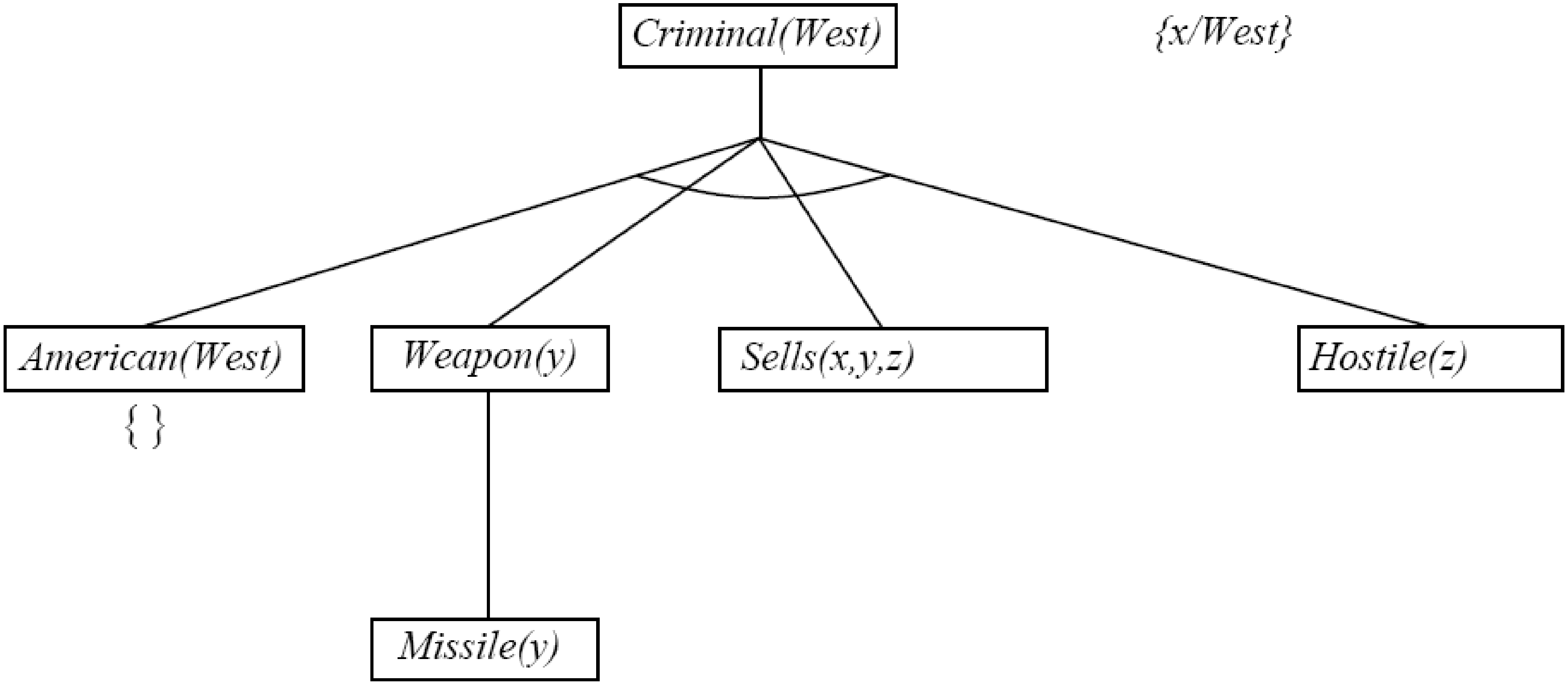
# Backward chaining example



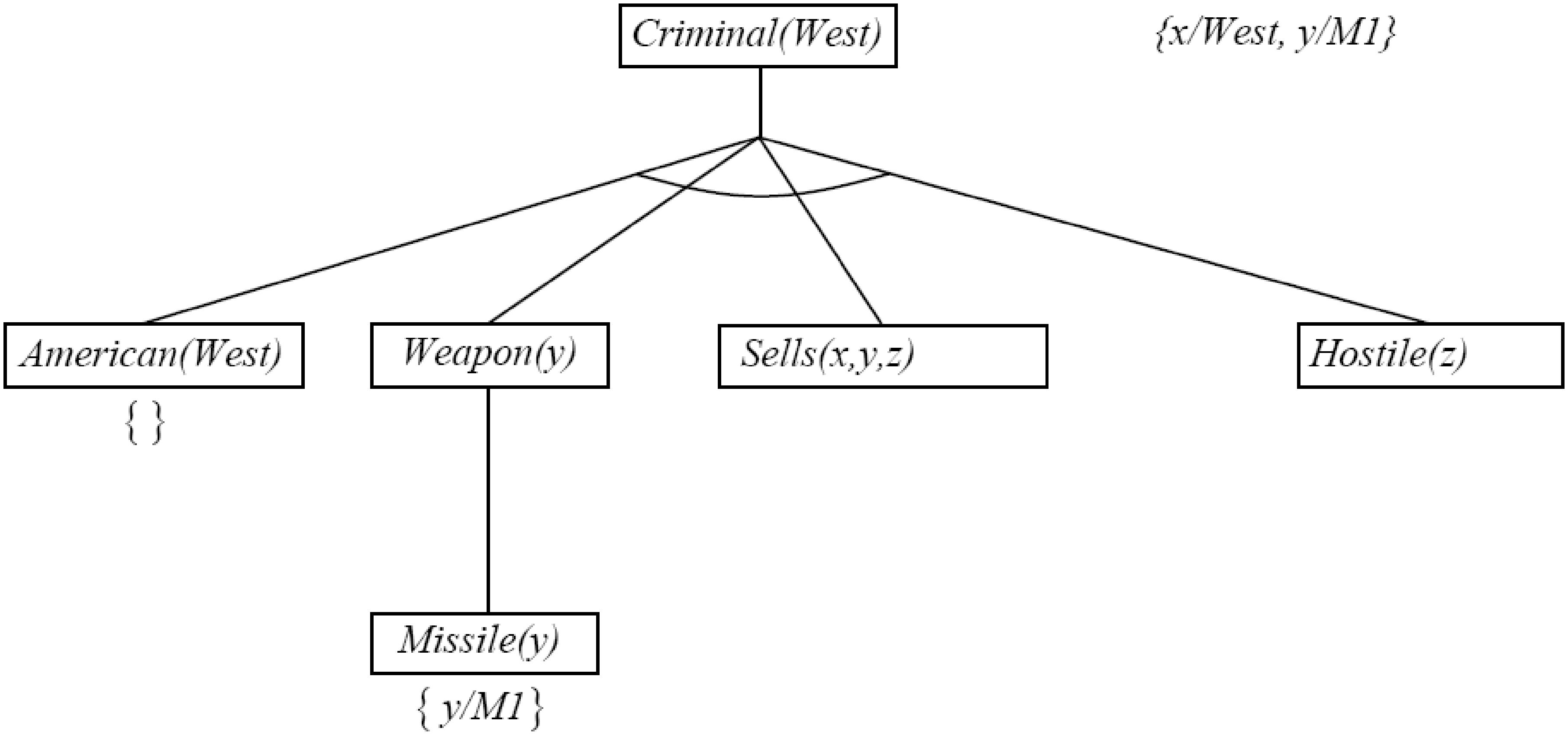
# Backward chaining example



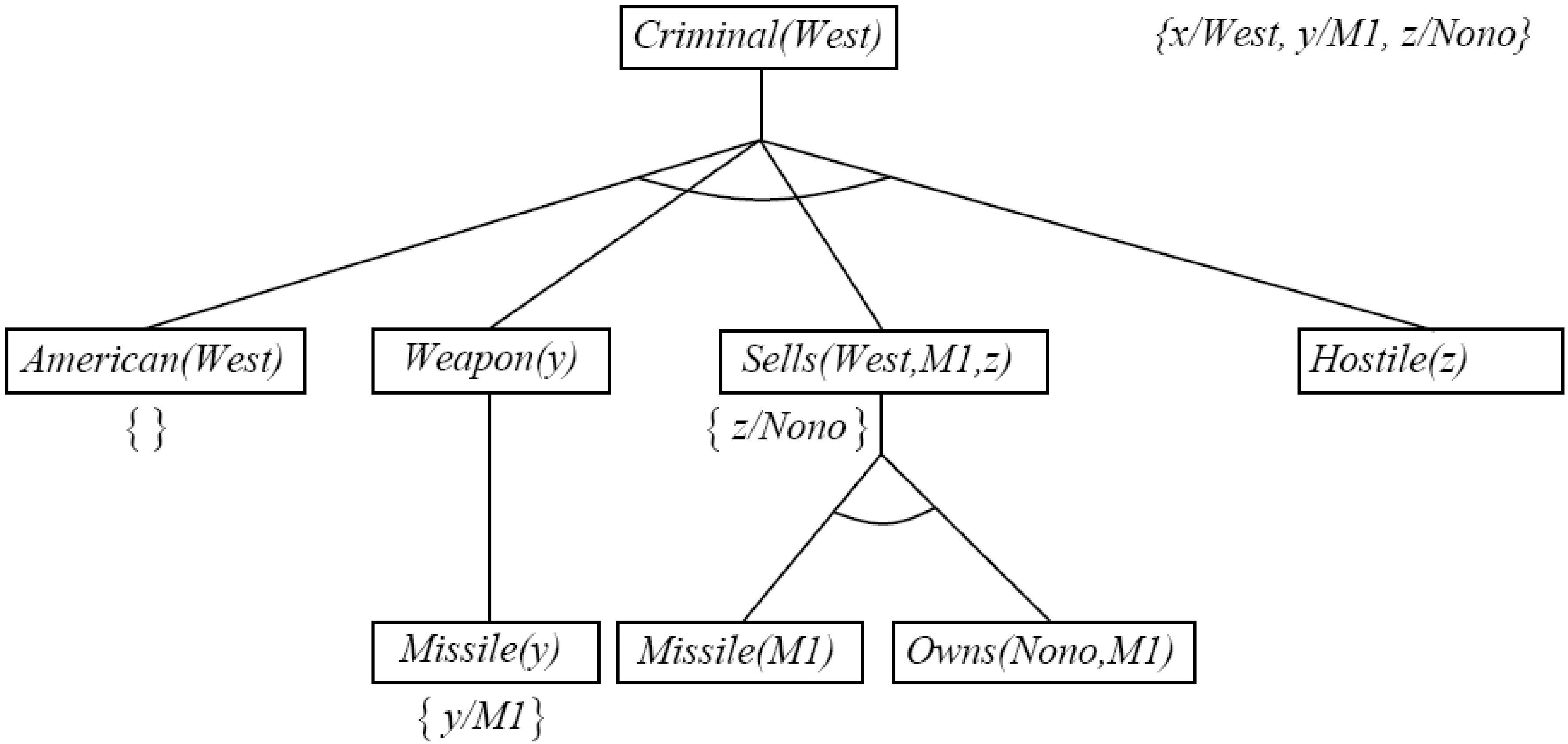
# Backward chaining example



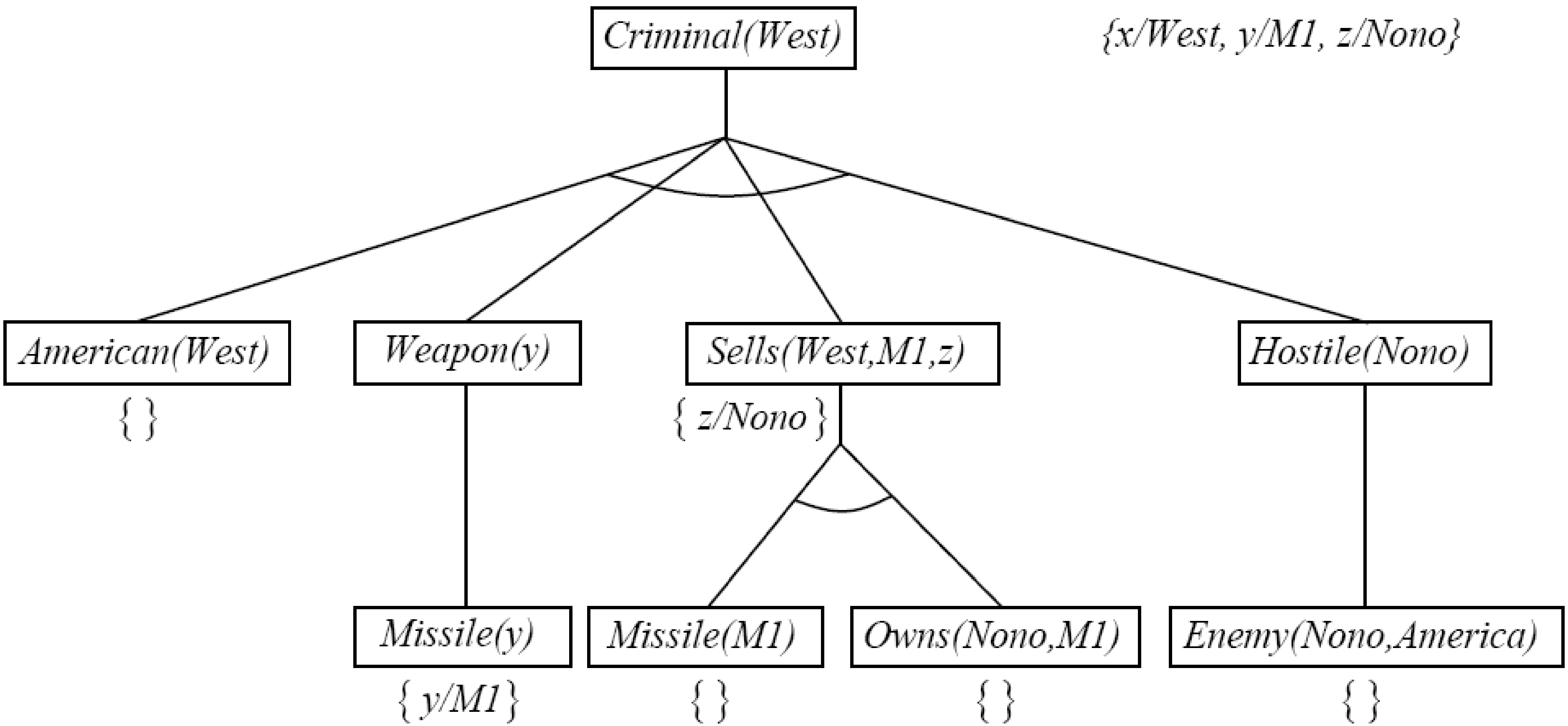
# Backward chaining example



# 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 (extra space!)

Widely used (without improvements!) for logic programming

# Logic programming

Sound bite: computation as inference on logical KBs

| | Logic programming | Ordinary programming |
|---|---|---|
| 1. | Identify problem | Identify problem |
| 2. | Assemble information | Assemble information |
| 3. | Tea break | Figure out solution |
| 4. | Encode information in KB | Program solution |
| 5. | Encode problem instance as facts | Encode problem instance as data |
| 6. | Ask queries | Apply program to data |
| 7. | Find false facts | Debug procedural errors |

Should be easier to debug $Capital(NewYork, US)$ than $x := x + 2$ !

# The End!