# Artificial Intelligence

First-Order Logic

Marc Toussaint
University of Stuttgart
Winter 2016/17

(slides based on Stuart Russell's AI course)

## Motivation:

First-order logic (FOL) is exactly what is sometimes been thought of as “Good Old-Fashioned AI” (GOFAI) – and what was the central target of critique on AI research coming from other fields like probabilistic reasoning and machine learning. A bit over-simplified, in the AI winter many researchers said “logic doesn’t work”, therefore AI doesn’t work, and instead the focus should be on learning and probabilistic modelling. Some comments on this:

First, I think one should clearly distinguish between 1) logic reasoning and inference, and 2) “first-order (or relational) representations”. Logic reasoning indeed is only applicable on discrete & deterministic knowledge bases. And as learnt knowledge is hardly deterministic (it cannot be in a Bayesian view), logic reasoning does not really apply well. In my view, this is one of the core problems with GOFAI: the fact that logic reasoning does not unify well with learning and learned models.

However, using “first-order (or relational) representations” means to represent knowledge in such a way that it refers only to object properties and relations, and therefore generalizes across object identities. Sure, classical FOL knowledge bases are first-order knowledge representations. But here research has advanced tremendously: nowadays we can also represent learned classifiers/regressions, graphical models, and Markov Decision Processes in a first-order (also called “relational” or “lifted”) way. The latter are core probabilistic formalisms to account for uncertainty and learning. Therefore the current state-of-the-art provides a series of unifications of probabilistic and first-order representations. I think this is what makes it important to learn and understand first-order representations – which is best taught in the context of FOL.

The reasoning and inference methods one requires for modern relational probabilistic models are of course different to classical logical reasoning. Therefore, I think knowing about “logical

reasoning" is less important than knowing about "logical representations". Still, some basic aspects of logical reasoning, such as computing all possible substitutions for an abstract sentence, thereby *grounding* the sentence, are essential in all first-order models.

Modern research on relational machine learning has, around 2011, lead to some new optimism about modern AI, also called the spring of AI (see, e.g., "I, algorithm: A new dawn for artificial intelligence", 2011). That wave of optimism now got over-rolled by the new hype on deep learning, which in the media is often equated with AI. However, at least up to now, one should clearly distinguish between deep learning as a great tool for machine learning with huge amounts of data; and reasoning, which includes model-based decision making, control, planning, and also (Bayesian) learning from few data.

This lecture introduces to FOL. The goal is to understand FOL as the basis for decision-making problems such as STRIPS rules, as well as for relational probabilistic models such as relational Reinforcement Learning and statistical relational learning methods. The latter are (briefly) introduced in the next lecture.

We first introduce the FOL language, then basic inference algorithms. Perhaps one of the most important concepts is the problem of computing substitutions (also called unification or matching problem), where much of the computational complexity of FOL representations arises.

# The FOL language

FOL is a language—we define the syntax, the semantics, and give examples.

# The limitation of propositional logic

- Propositional logic has nice properties:
  - 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)

- Limitation:
  - Propositional logic has very limited expressive power, unlike natural language. E.g., we cannot express "pits cause breezes in adjacent squares" except by writing one sentence for each square

# First-order logic

- Whereas propositional logic assumes that a 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 . . .

# FOL syntax elements

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

# FOL syntax grammar

$$
\begin{array}{lcl}
\langle \text{sentence} \rangle & \rightarrow & \langle \text{atomic sentence} \rangle \\
 & & \mid \langle \text{complex sentence} \rangle \\
 & & \mid [\forall \mid \exists] \ \langle \text{variable} \rangle \ \ \langle \text{sentence} \rangle \\
 & & \\
\langle \text{atomic sentence} \rangle & \rightarrow & \text{predicate}(\langle \text{term} \rangle, \ldots) \\
 & & \mid \langle \text{term} \rangle = \langle \text{term} \rangle \\
 & & \\
\langle \text{term} \rangle & \rightarrow & \text{function}(\langle \text{term} \rangle, \ldots) \\
 & & \mid \text{constant} \\
 & & \mid \text{variable} \\
 & & \\
\langle \text{complex sentence} \rangle & \rightarrow & \neg \ \langle \text{sentence} \rangle \\
 & & \mid (\langle \text{sentence} \rangle \ [\wedge \mid \vee \mid \ \Rightarrow \ \mid \ \Leftrightarrow \ ] \ \langle \text{sentence} \rangle)
\end{array}
$$

# Quantifiers

- Universal quantification

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

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

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

- Existential quantification

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

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

Example: "Someone at Stanford is smart:" $\exists x \ \ At(x, Stanford) \wedge Smart(x)$

# 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

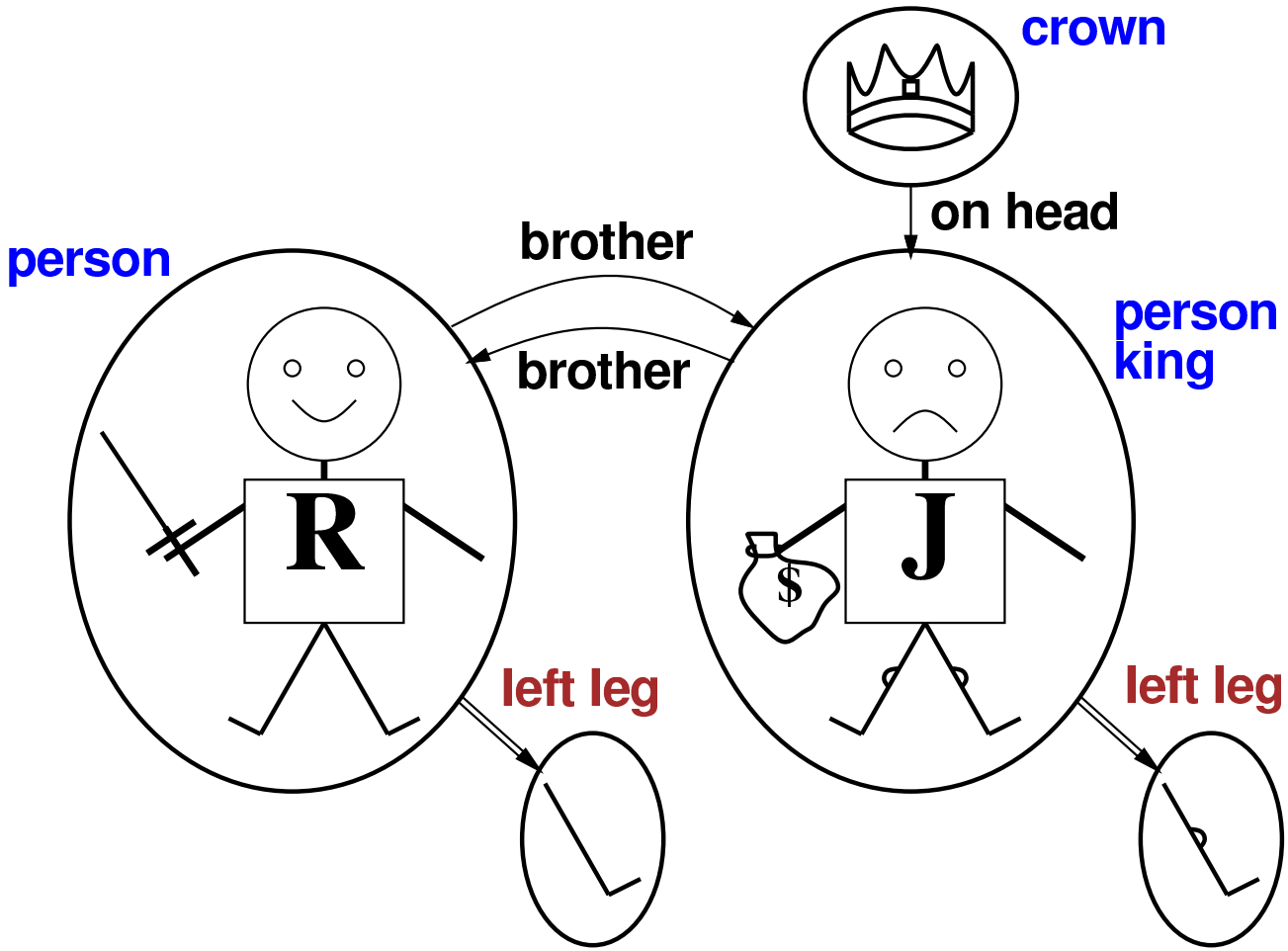
  $\forall x \ Likes(x, IceCream) \quad \equiv \quad \neg\exists x \ \neg Likes(x, IceCream)$

  $\exists x \ Likes(x, Broccoli) \quad \equiv \quad \neg\forall x \ \neg Likes(x, Broccoli)$

# Truth in first-order logic

- Sentences are true with respect to a model and an interpretation
- A model contains $\geq 1$ objects and relations among them
- An 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



# Models for FOL: Lots!

- Entailment in propositional logic can be computed by enumerating models
- We can also enumerate the FOL models for a given KB:
  - 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 ...
- Enumerating FOL models is very inefficient

# Example sentences

- “Brothers are siblings”

# Example sentences

- “Brothers are siblings”

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

- “Sibling” is symmetric

# Example 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"

# Example 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"

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

# FOL Inference

# Universal instantiation (UI)

- Whenever a KB contains a universally quantified sentence, we may add to the KB any instantiation of that sentence, where the logic variable $v$ is replaced by a concrete ground term $g$:

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

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

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

# Existential instantiation (EI)

- Whenever a KB contains a existentially quantified sentence $\exists v \;\; \alpha$, we may add a single instantiation of that sentence to the KB, where the logic variable $v$ is replaced by a Skolem constant symbol $k$ *which must 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

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

where $e$ is a new constant symbol

# Instantiations 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

- Instantiating all quantified sentences allows us to ground the KB, that is, to make the KB propositional
- Example: 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.

# Theory on propositionalization

- Claim: A ground sentence$^*$ is entailed by the propositionalized KB iff entailed by original FOL KB
  (or "Every FOL KB can be propositionalized so as to preserve entailment")
- Then, FOL inference can be done by: propositionalize KB and query, apply resolution, return result
- Problem: with *function* symbols, there are infinitely many ground terms, e.g., $Father(Father(Father(John)))$
- Theorem: Herbrand (1930). If a sentence $\alpha$ is entailed by an FOL KB, it is entailed by a *finite* subset of the propositional KB
- Idea: For $n = 0$ to $\infty$ do
  create a propositional KB by instantiating with depth-$n$ terms
  see if $\alpha$ is entailed by this KB
- Problem: works if $\alpha$ is entailed, loops if $\alpha$ is not entailed
- Theorem: Turing (1936), Church (1936), entailment in FOL is semidecidable

# Inefficiency of naive propositionalization

- Propositionalization generates lots of irrelevant sentences.
  Example:

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

  propositionalization produces not only $Greedy(John)$, but also $Greedy(Richard)$ which is irrelevant for a query $Evil(John)$

- With $p$ $k$-ary predicates and $n$ constants, there are $p \cdot n^k$ instantiations
  With function symbols, it gets much much worse!

# Unification

- Instead of instantiating quantified sentences in all possible ways, we can compute specific substitutions "that make sense". These are substitutions that *unify* abstract sentences so that rules (Horn clauses, GMP, see next slide) can be applied.
- In the previous example, the "Evil-rule" can be applied if we can find a substitution $\theta$ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$. Namely, $\theta = \{x/John, y/John\}$ is such a substitutions.

  We write $\theta\,\text{unifies}(\alpha, \beta)$ iff $\alpha\theta = \beta\theta$
- Examples:

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | |

# Unification

- Instead of instantiating quantified sentences in all possible ways, we can compute specific substitutions "that make sense". These are substitutions that *unify* abstract sentences so that rules (Horn clauses, GMP, see next slide) can be applied.
- In the previous example, the "Evil-rule" can be applied if we can find a substitution $\theta$ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$. Namely, $\theta = \{x/John, y/John\}$ is such a substitutions.

  We write $\theta$ unifies$(\alpha, \beta)$ iff $\alpha\theta = \beta\theta$
- Examples:

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

# Unification

- Instead of instantiating quantified sentences in all possible ways, we can compute specific substitutions "that make sense". These are substitutions that *unify* abstract sentences so that rules (Horn clauses, GMP, see next slide) can be applied.
- In the previous example, the "Evil-rule" can be applied if we can find a substitution $\theta$ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$. Namely, $\theta = \{x/John, y/John\}$ is such a substitutions.

  We write $\theta\,\text{unifies}(\alpha, \beta)$ iff $\alpha\theta = \beta\theta$
- Examples:

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

# Unification

- Instead of instantiating quantified sentences in all possible ways, we can compute specific substitutions "that make sense". These are substitutions that *unify* abstract sentences so that rules (Horn clauses, GMP, see next slide) can be applied.
- In the previous example, the "Evil-rule" can be applied if we can find a substitution $\theta$ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$. Namely, $\theta = \{x/John, y/John\}$ is such a substitutions.

  We write $\theta$ unifies$(\alpha, \beta)$ iff $\alpha\theta = \beta\theta$
- Examples:

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

# Unification

- Instead of instantiating quantified sentences in all possible ways, we can compute specific substitutions "that make sense". These are substitutions that *unify* abstract sentences so that rules (Horn clauses, GMP, see next slide) can be applied.
- In the previous example, the "Evil-rule" can be applied if we can find a substitution $\theta$ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$. Namely, $\theta = \{x/John, y/John\}$ is such a substitutions.

  We write $\theta$ unifies$(\alpha, \beta)$ iff $\alpha\theta = \beta\theta$
- Examples:

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

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

# Generalized Modus Ponens (GMP)

- For every substitution $\theta$ such that $\forall_i : \theta \,\text{unifies}(p_i', p_i)$ we can apply:

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

  Example:

  $p_1'$ is $King(John)$ $\quad$ $p_1$ is $King(x)$

  $p_2'$ is $Greedy(y)$ $\quad$ $p_2$ is $Greedy(x)$

  $\theta$ is $\{x/John, y/John\}$ $\quad$ $q$ is $Evil(x)$

  $q\theta$ is $Evil(John)$

- This GMP assumes a KB of definite clauses (*exactly* one positive literal)
  By default, all variables are assumed universally quantified.

# Forward chaining algorithm

**function** FOL-FC-ASK($KB, \alpha$) **returns** a substitution or *false*

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

## Example: Crime

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: Crime – formalization

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

# Example: Crime – formalization

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

# Example: Crime – formalization

- . . . 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

# Example: Crime – formalization

- . . . 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:

# Example: Crime – formalization

- . . . 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”:

# Example: Crime – formalization

- . . . 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 . . .

# Example: Crime – formalization

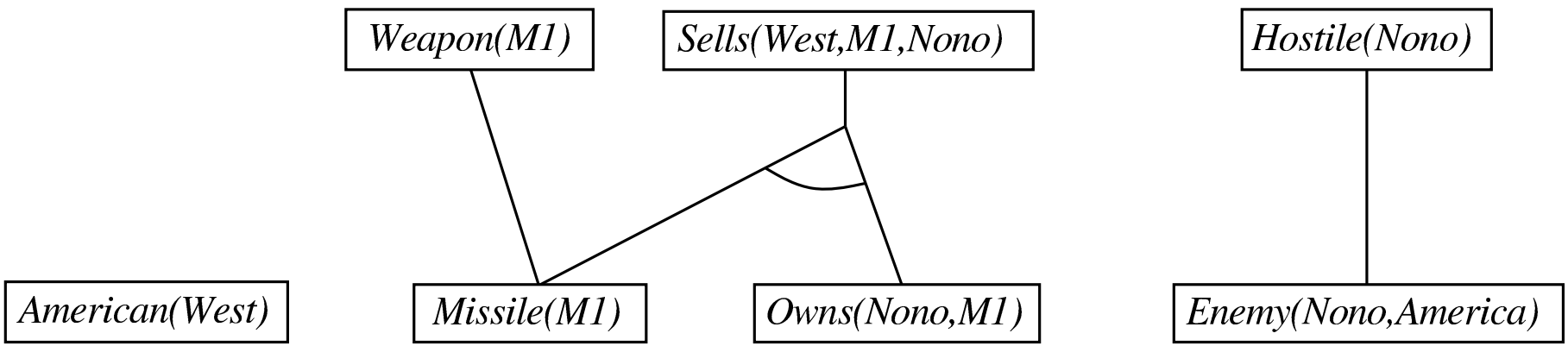
- . . . it is a crime for an American to sell weapons to hostile nations:
  $American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$
- Nono . . . has some missiles, i.e., $\exists x\, Owns(Nono, x) \land Missile(x)$:
  $Owns(Nono, M_1)$ and $Missile(M_1)$
- . . . all 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 . . .

# Example: Crime – formalization

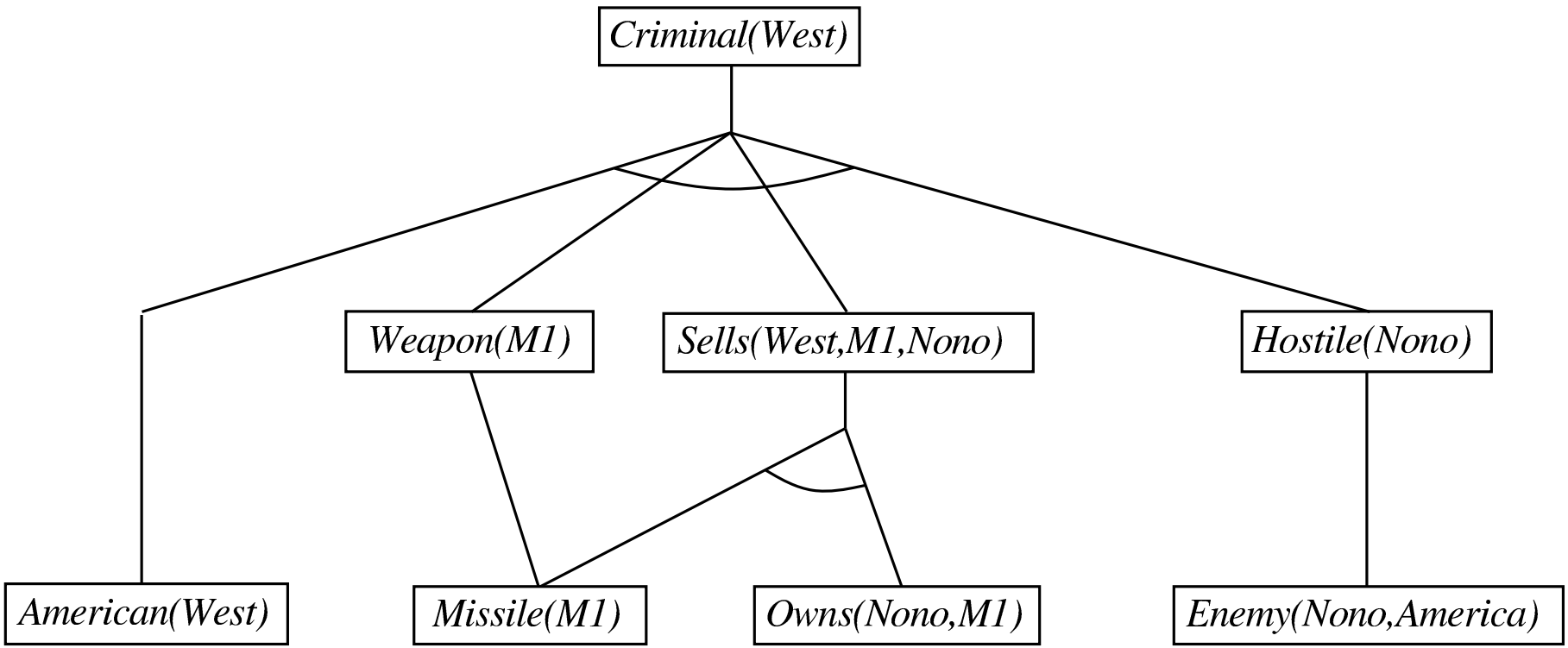
- . . . it is a crime for an American to sell weapons to hostile nations:
  $American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$
- Nono . . . has some missiles, i.e., $\exists x\, Owns(Nono, x) \land Missile(x)$:
  $Owns(Nono, M_1)$ and $Missile(M_1)$
- . . . all 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 . . .
  $Enemy(Nono, America)$

# Example: Crime – forward chaining proof

# Example: Crime – forward chaining proof



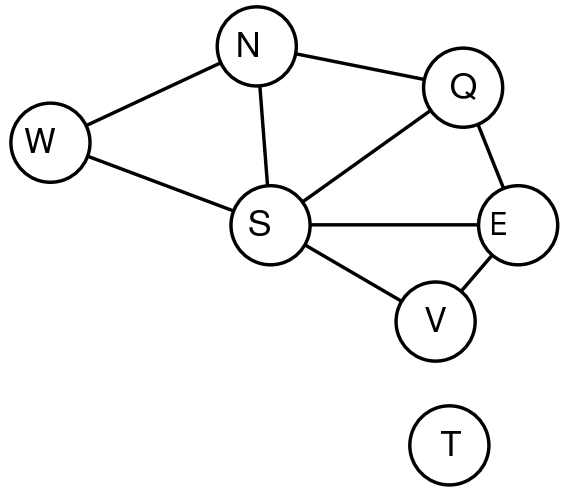
# Example: Crime – forward chaining proof



# Properties of forward chaining

- Sound and complete for first-order definite clauses (proof similar to propositional proof)
- Datalog = first-order definite clauses + *no functions* (e.g., crime KB).
  Forward chaining terminates for Datalog in poly iterations: at most $p \cdot n^k$ literals
- May not terminate in general if $\alpha$ is not entailed
  This is unavoidable: entailment with definite clauses is semidecidable
- Efficiency:
  - Simple observation: no need to match (=compute possible substitutions) a rule on iteration $k$ if a premise wasn't added on iteration $k-1$ $\Rightarrow$ match only rules whose premise contain a newly added literal
  - Matching (computing substitutions) can be expensive:
    - Database indexing allows $O(1)$ retrieval of known facts, e.g., query $Missile(x)$ retrieves $Missile(M_1)$
    - But matching conjunctive premises against known facts is NP-hard (is a CSP problem, see below)

# Hard matching example: a CSP




- Consider the KB:

$$\mathit{Diff}(wa, nt) \land \mathit{Diff}(wa, sa) \land \mathit{Diff}(nt, q) \land \mathit{Diff}(nt, sa) \land$$
$$\mathit{Diff}(q, nsw) \land \mathit{Diff}(q, sa) \land \mathit{Diff}(nsw, v) \land \mathit{Diff}(nsw, sa) \land$$
$$\mathit{Diff}(v, sa) \Rightarrow Colorable()$$
$$\mathit{Diff}(Red, Blue), \quad \mathit{Diff}(Red, Green), \quad \mathit{Diff}(Green, Red)$$
$$\mathit{Diff}(Green, Blue), \quad \mathit{Diff}(Blue, Red), \quad \mathit{Diff}(Blue, Green)$$

- $Colorable()$ is inferred iff the CSP has a solution
  CSPs include 3SAT as a special case, hence matching is NP-hard

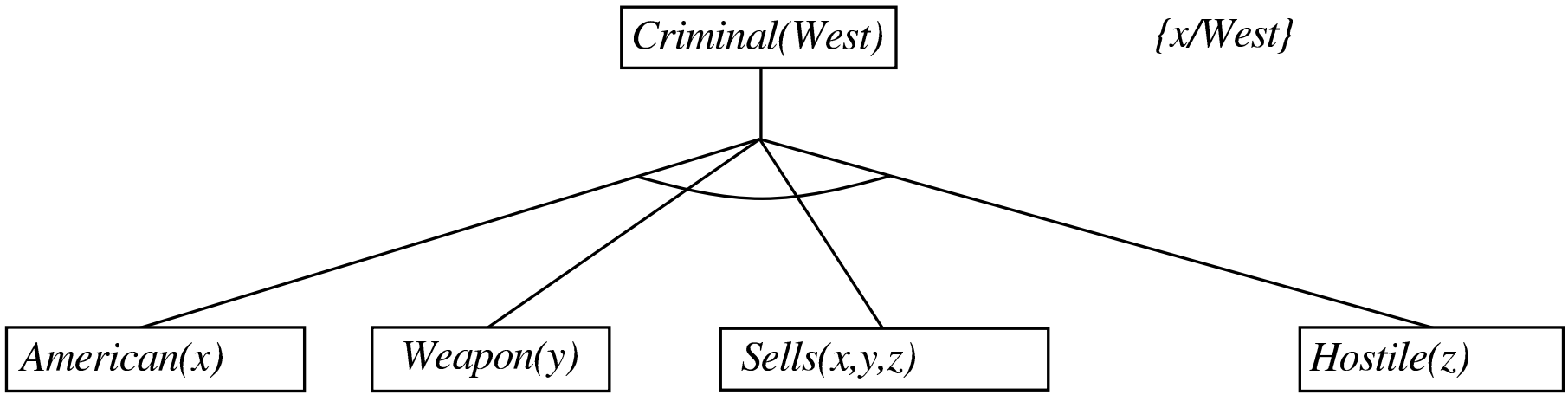
# Backward chaining algorithm*

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

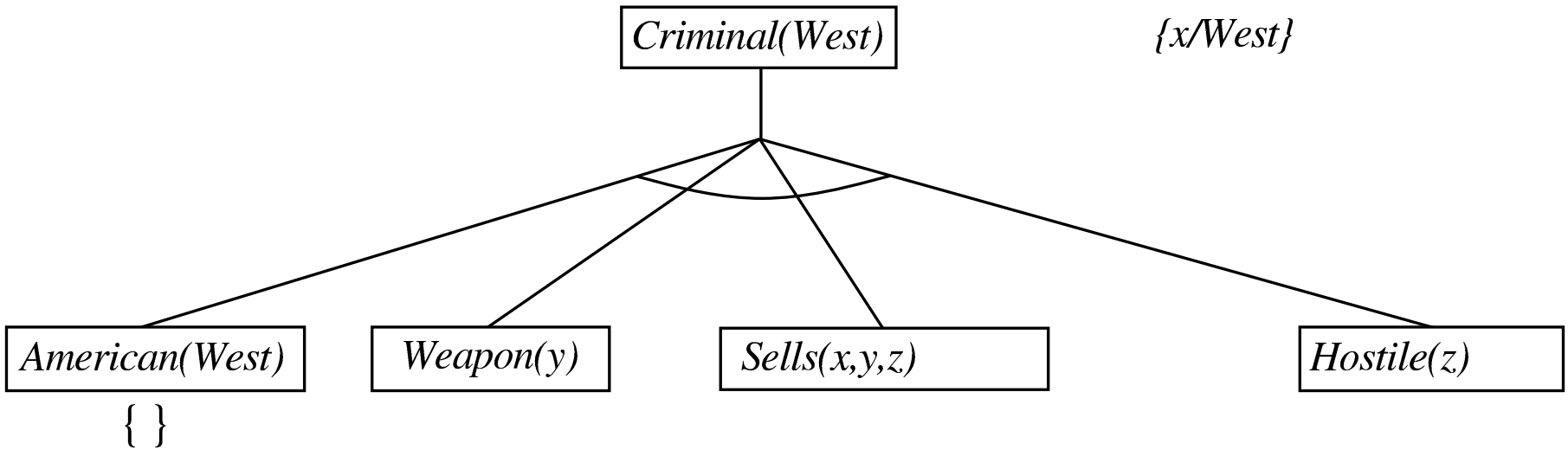
  **if** *goals* is empty **then return** $\{\theta\}$
  $q' \leftarrow$ SUBST($\theta$, FIRST(*goals*))
  **for each** sentence $r$ **in** $KB$
    where STANDARDIZE-APART($r$) = $(p_1 \wedge \ldots \wedge p_n \Rightarrow q)$
    and $\theta' \leftarrow$ UNIFY($q$, $q'$) succeeds
    *new_goals* $\leftarrow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

# Backward chaining example*

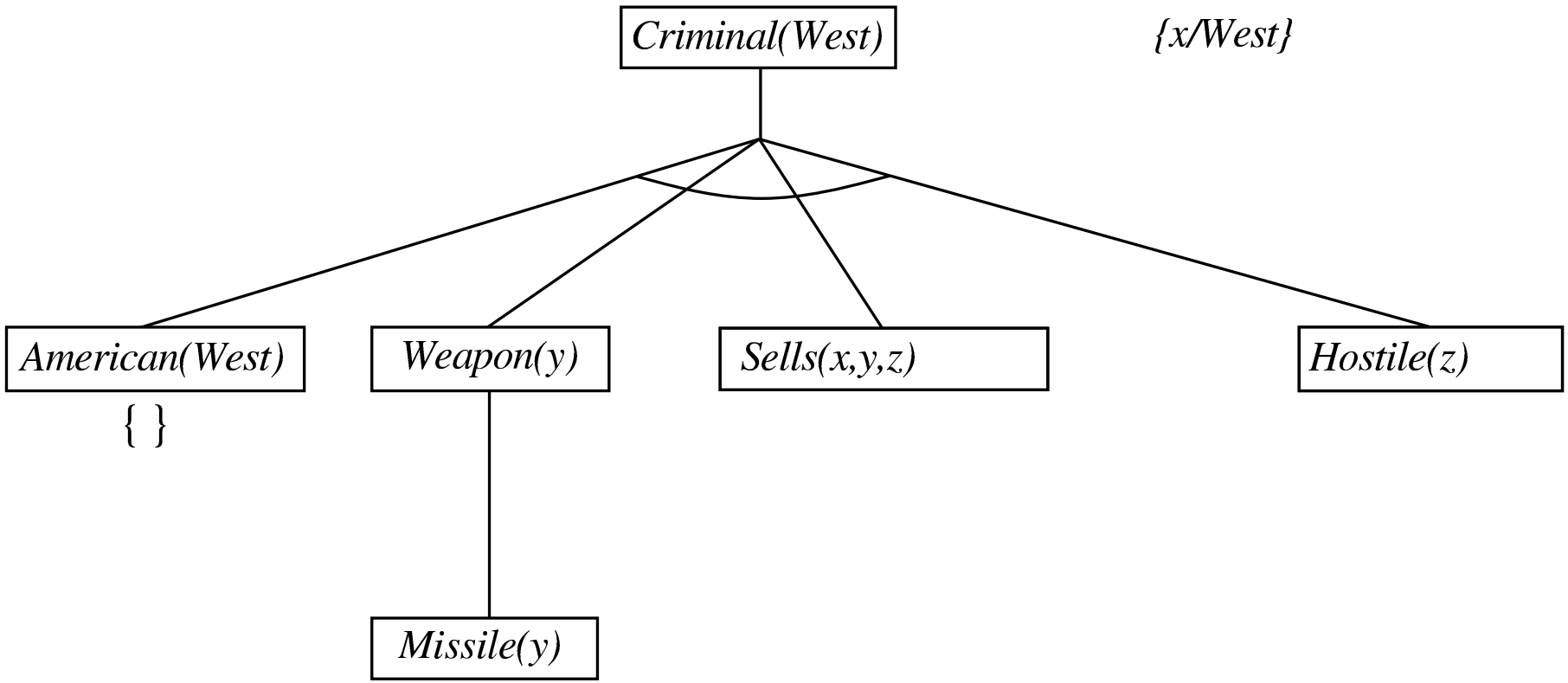
# Backward chaining example*



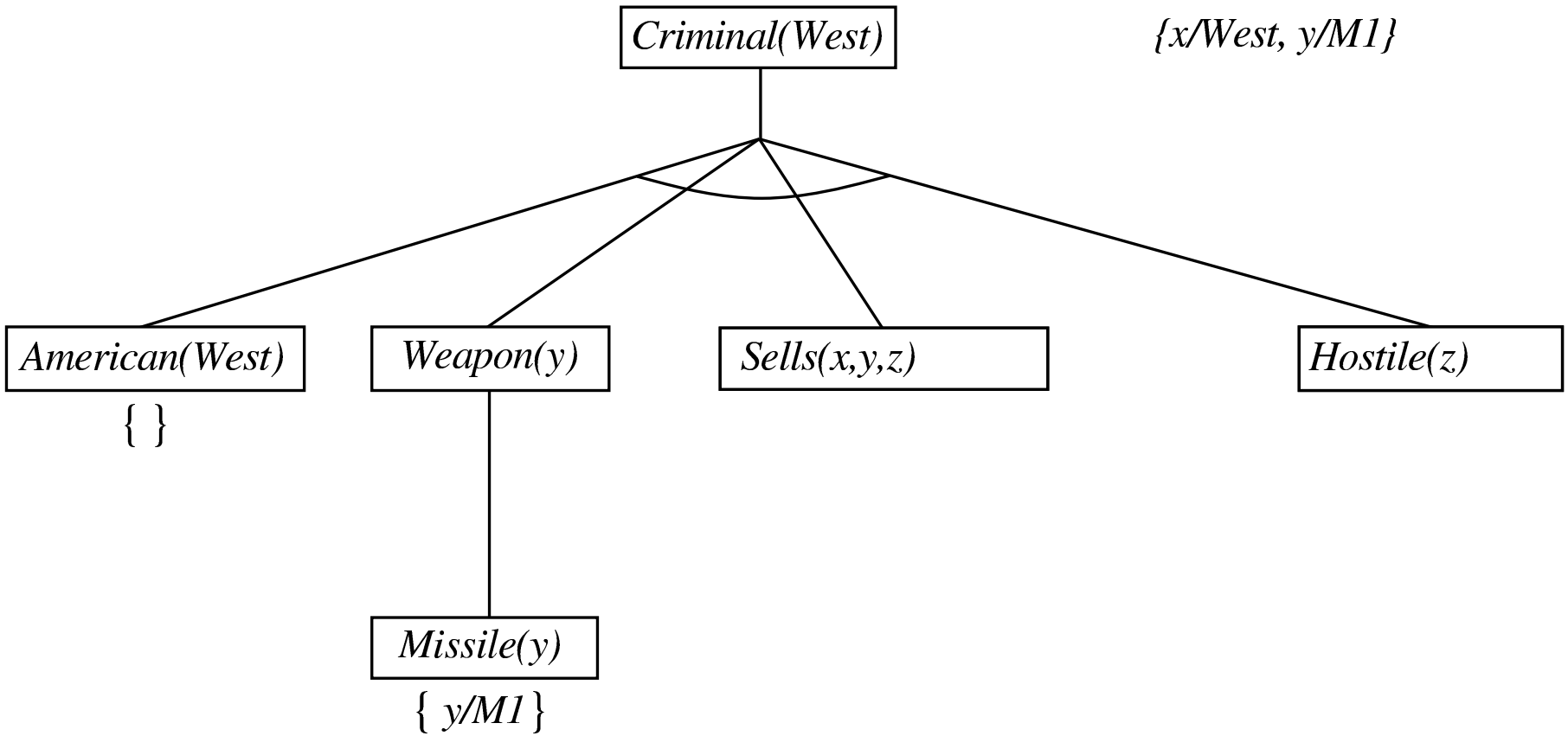
# Backward chaining example*



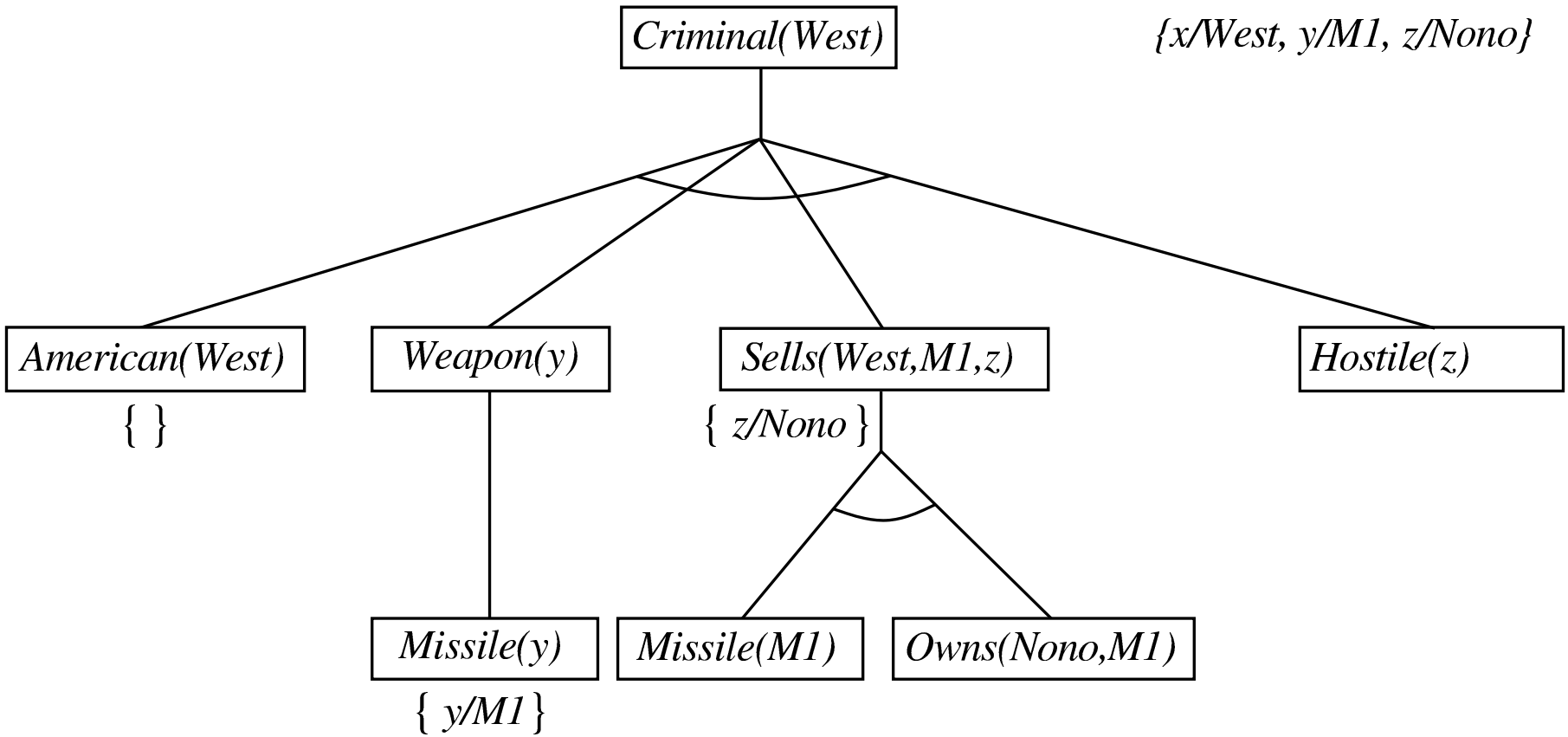
# Backward chaining example*



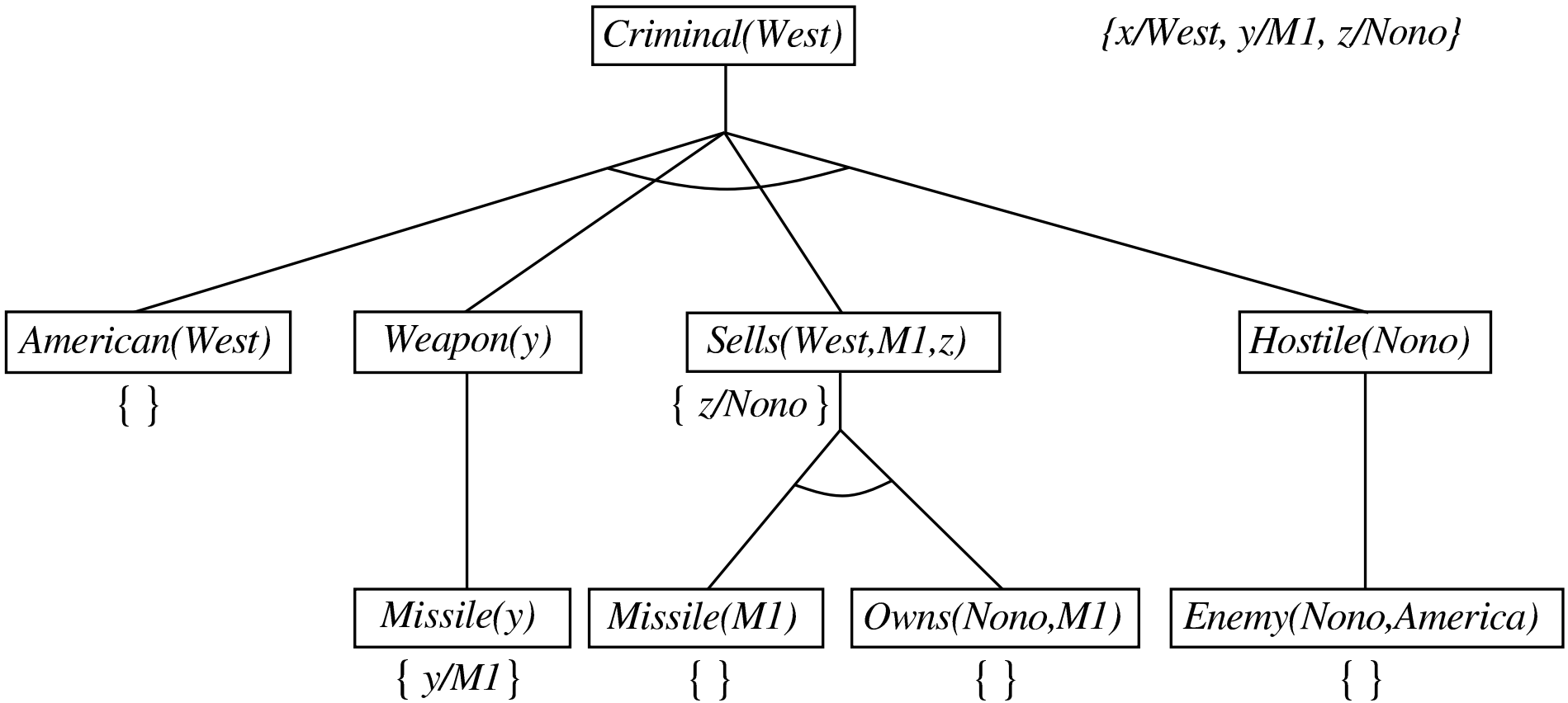
# Backward chaining example*



# 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 (extra space!)
- Widely used (without improvements!) for logic programming

# Example: Prolog*

- Declarative vs. imperative programming:

| | 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 |

- Russell says "should be easier to debug $Capital(NewYork, US)$ than $x := x + 2$!"...

# Prolog systems*

- Basis: backward chaining with Horn clauses + bells & whistles
  Widely used in Europe, Japan (basis of 5th Generation project)
  Compilation techniques $\Rightarrow$ approaching a billion LIPS
- Program = set of clauses `head :- literal`$_1$`, ... literal`$_n$`.`

  ```
  criminal(X) :- american(X), weapon(Y), sells(X,Y,Z), hostile(Z).
  ```
- Closed-world assumption ("negation as failure")
  e.g., given `alive(X) :- not dead(X).`
  `alive(joe)` succeeds if `dead(joe)` fails
- Details:
  - Efficient unification by open coding
  - Efficient retrieval of matching clauses by direct linking
  - Depth-first, left-to-right backward chaining
  - Built-in predicates for arithmetic etc., e.g., `X is Y*Z+3`

# Prolog examples*

- Depth-first search from a start state `X`:

```
dfs(X) :- goal(X).
dfs(X) :- successor(X,S),dfs(S).
```

  No need to loop over `S`: `successor` succeeds for each

- Appending two lists to produce a third:

```
append([],Y,Y).
append([X|L],Y,[X|Z]) :- append(L,Y,Z).

query:   append(A,B,[1,2]) ?
answers: A=[]    B=[1,2]
         A=[1]   B=[2]
         A=[1,2] B=[]
```

# Conversion to CNF

Everyone who loves all animals is loved by someone:

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

1. Eliminate biconditionals and implications

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

2. 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.

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

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

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

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

5. Drop universal quantifiers:

$$[Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x)$$

6. Distribute $\land$ over $\lor$:

$$[Animal(F(x)) \lor Loves(G(x), x)] \land [\neg Loves(x, F(x)) \lor Loves(G(x), x)]$$

# Resolution: brief summary

- For any substitution $\theta$ $\text{unifies}(\ell_i, \neg m_j)$ for some $i$ and $j$, apply:

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

  Example:

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

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

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

# Example: crime – resolution proof