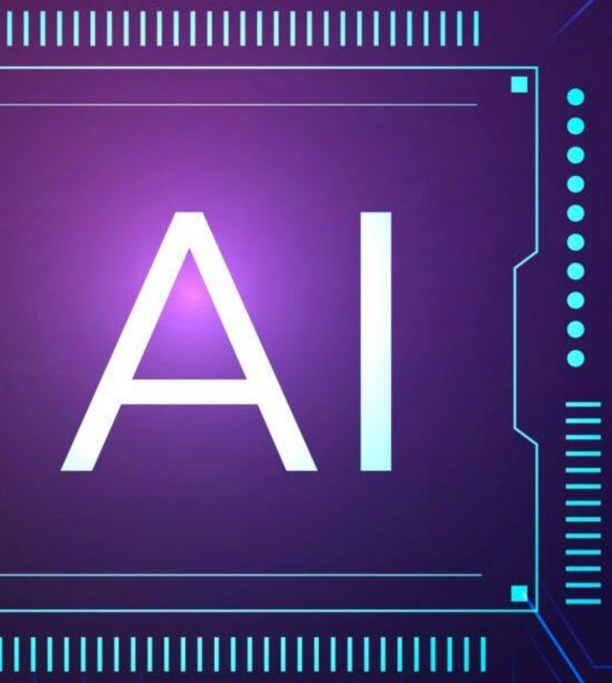


FIRST-ORDER LOGIC

Nguyễn Ngọc Thảo – Nguyễn Hải Minh
{nnthao, nhminh}@fit.hcmus.edu.vn

Outline

- Representation revisited
- First-order logic (FOL): Syntax and semantics
- Writing FOL sentences: Case studies
- First-order inference
    - Unification and lifting
    - Forward chaining and Backward chaining
    - Resolution

Representation revisited

Programming languages

- Programming languages use data structures to define facts and programs to express computational processes.
  - E.g., implement the Wumpus world as a 4×4 array, "$\mathrm{World}[2,2] \leftarrow \mathrm{Pit}$" means "There is a pit in square [2,2]."
- They lack mechanisms to derive new facts from other ones.
  - Update to a data structure is done by a domain-specific procedure
- They lack expressiveness to handle partial information.
  - E.g., to say "There is a pit in [2,2] or [3,1]", a program stores a single value for each variable and allows the value to be "unknown"
  - Meanwhile, the PL-sentence, $P_{2,2} \vee P_{3,1}$, is more intuitive

Propositional logic (PL)

- In propositional logic, knowledge and inference are separate, with inference being domain-independent.
- Context-independent: A sentence's meaning depends on the meaning of its parts.
    - E.g., The meaning of $S_{1,4} \wedge S_{1,2}$ relates the meanings of $S_{1,4}$ and $S_{1,2}$.
- Limited expressive power
    - E.g., "Squares adjacent to pits are breezy." is expressed by writing one sentence for each square,

$$B_{1,1} \Leftrightarrow \left(P_{1,2} \vee P_{2,1}\right),\ B_{2,2} \Leftrightarrow \left(P_{1,2} \vee P_{2,1} \vee P_{3,2} \vee P_{3,1}\right), \text{etc.}$$

First-order logic (FOL)

- The syntax and semantics of first-order logic is built around objects and relations.
- It can express facts about some or all objects in the universe.

Objects

- Referred by nouns and noun phrases
- E.g., people, numbers, colors, Joe, games, etc.

Relations

- Referred by verbs and verb phrases
- Unary relation (properties): red, prime, etc.
- n-ary relation: brother of, comes between, etc.

Functions

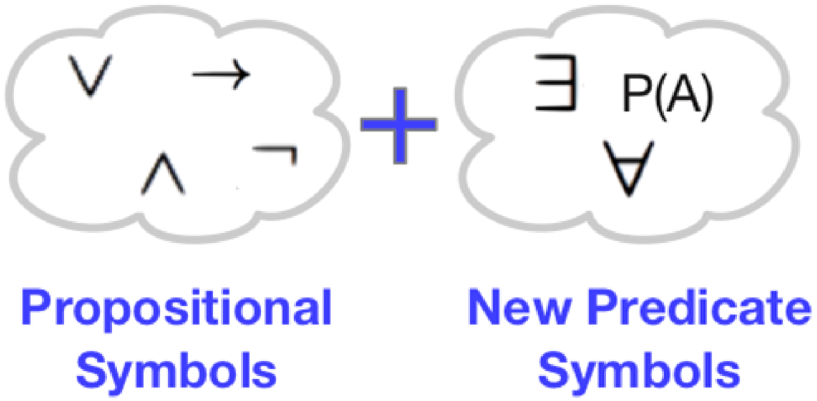
- Relations that have only one “value” for an “input”
- E.g., father of, best friend, sum of, etc.

First-order logic: Some examples

- "One plus two equals three."
  - Object: one, two, three, one plus two
  - Relation: equal
  - Function: plus
- "Squares neighboring the Wumpus are smelly."
  - Object: squares, Wumpus
  - Relation: neighboring
  - Property: smelly
- "Evil King John ruled England in 1200."
  - Objects: John, England, 1200
  - Relation: ruled during
  - Property: evil, king

Types of logics

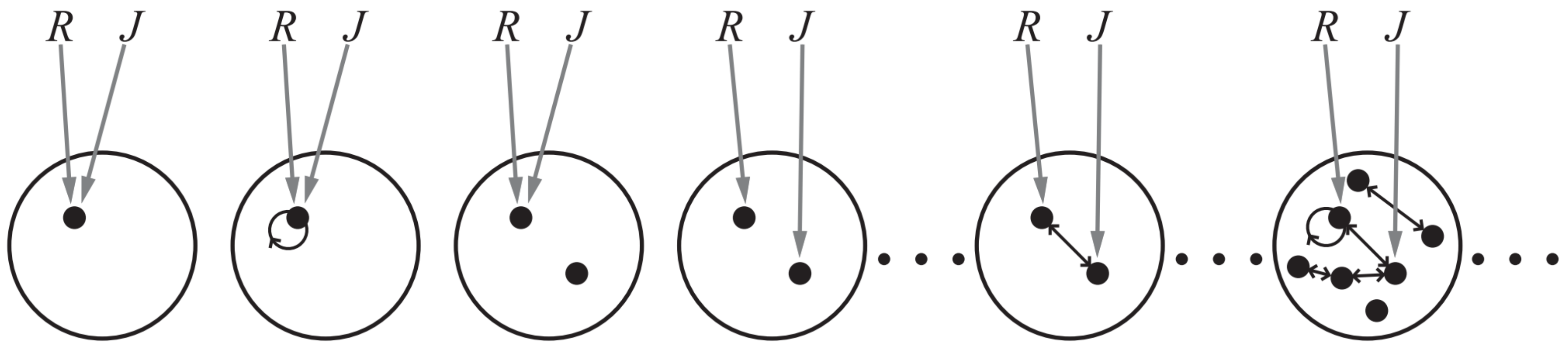
Language	Ontological Commitment (What exists in the world)	Epistemological Commitment (What an agent believes about facts)
Propositional logic	Facts	True/false/unknown
First-order logic	Facts, objects, relations	True/false/unknown
Temporal logic	Facts, objects, relations, time	True/false/unknown
Probability logic	Facts	Degree of belief $\in [0,1]$
Fuzzy logic	Facts with degree of truth $\in [0,1]$	Known interval value



FOL Syntax and Semantics

Possible models in First-order Logic

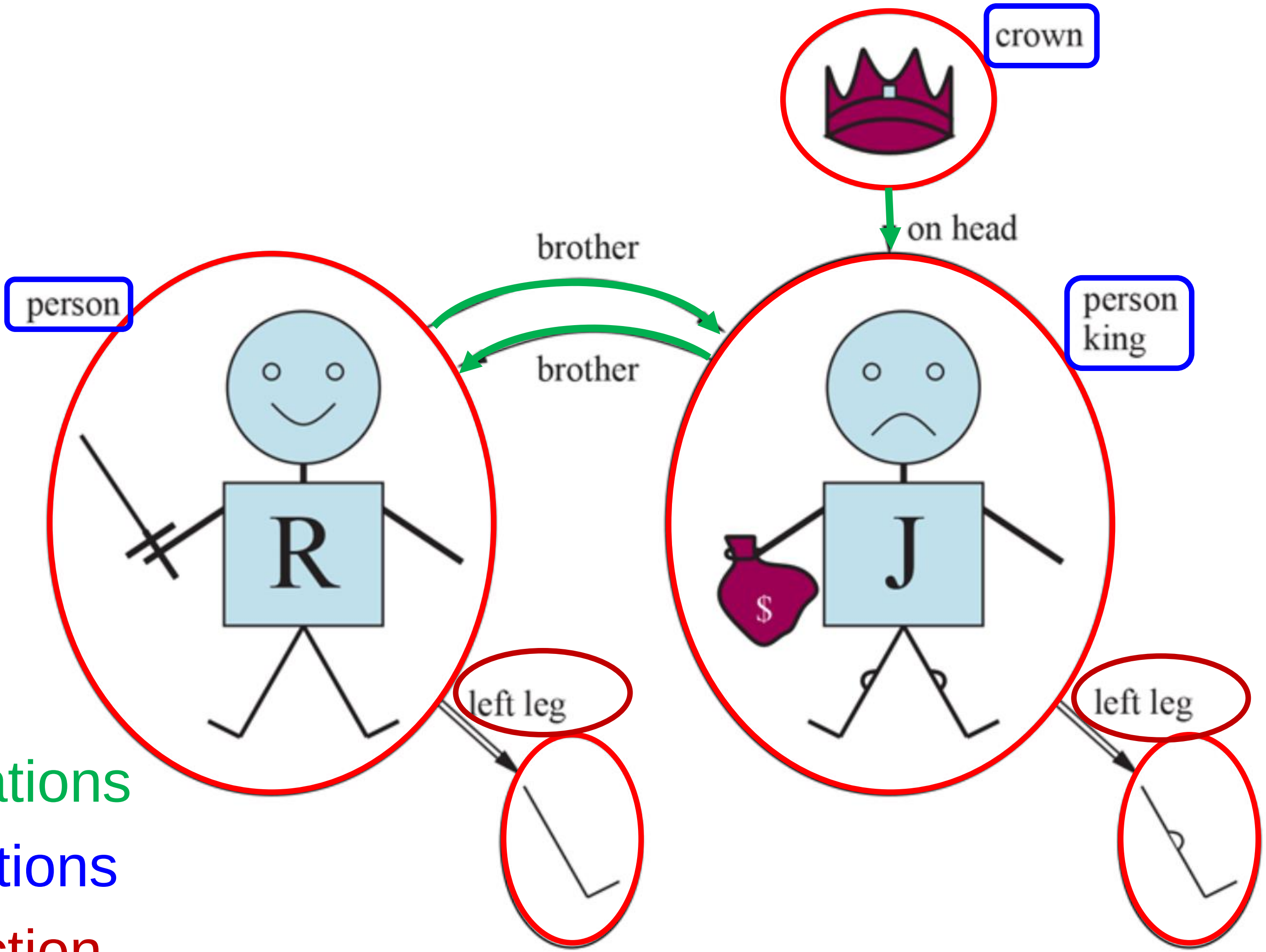
- Entailment, validity, and related concepts are defined based on all possible models.
  - A FOL-model is a set of objects (domain) and relations among them.



137,506,194,466 models with six or fewer objects.

- Entailment checking by the enumeration is **infeasible** due to the unbounded number of possible models.

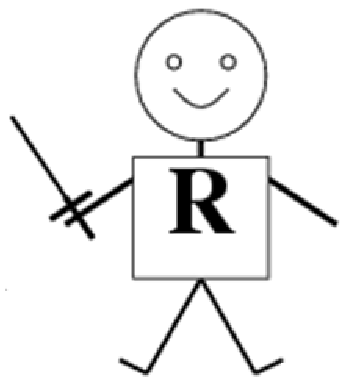
Models for FOL: A concrete example



- 5 objects
- 2 binary relations
- 3 unary relations
- 1 unary function

Models for FOL: A concrete example

- 5 objects

				
Richard the Lionheart (King of England 1189-1199)	King John (King of England 1199-1215)	The left leg of Richard	The left leg of John	A crown

Models for FOL: A concrete example

- Binary relations
  - The brotherhood relation

$\{ \langle$Richard the Lionheart, King John$\rangle$, $\langle$King John, Richard the Lionheart$\rangle \}$

  - The "on head" relation

    $\{ \langle$The crown, King John$\rangle \}$

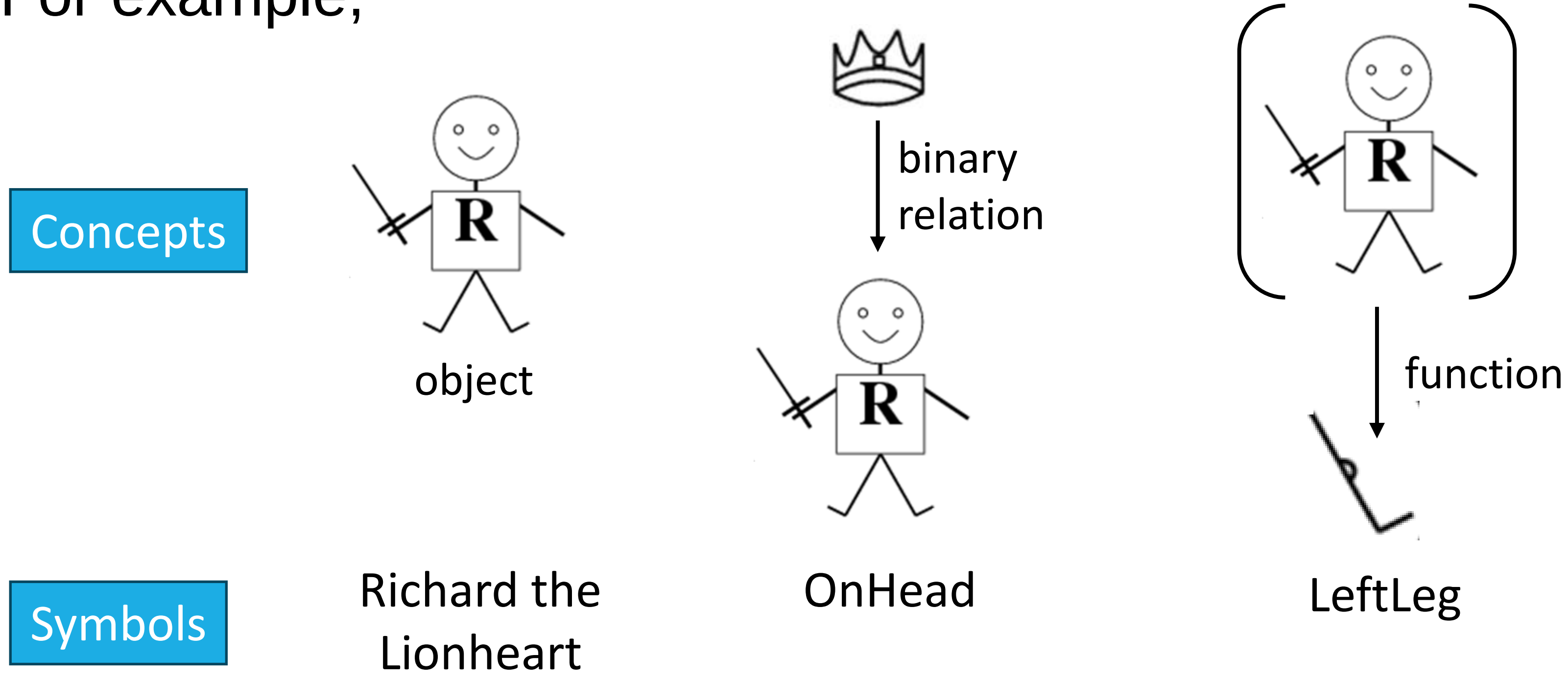
- Unary relations: "person", "king", "crown"
- Functions: "left leg"
  - $\langle$Richard the Lionheart$\rangle \rightarrow$ Richard's left leg
  - $\langle$King John$\rangle \rightarrow$ John's left leg

FOL concepts: Symbols

- Constant symbols represents objects.
  - E.g., Richard, John, etc.
- Predicate symbols stand for relations.
  - E.g., Brother , OnHead, Person, King, and Crown, etc.
- Function symbols stand for functions.
  - E.g., LeftLeg
- Each predicate or function symbol comes with an arity that fixes the number of arguments.
  - E.g., Brother(x,y) $\rightarrow$ binary, LeftLeg(x) $\rightarrow$ unary, etc.
- These symbols begins with uppercase letters by convention.

Symbols and intended interpretation

- **Interpretation** specifies exactly which objects, relations and functions are referred to by the symbols.
- Each model includes an (intended) interpretation.
- For example,



$$
\begin{array}{rcl}
\mathit{Sentence} & \rightarrow & \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} & \rightarrow & \mathit{Predicate} \mid \mathit{Predicate}(\mathit{Term}, \ldots) \mid \mathit{Term} = \mathit{Term} \\
\mathit{ComplexSentence} & \rightarrow & (\ \mathit{Sentence}\ ) \\
 & \mid & \neg\, \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \land \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \lor \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \ \Rightarrow\ \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \ \Leftrightarrow\ \mathit{Sentence} \\
 & \mid & \mathit{Quantifier}\ \mathit{Variable}, \ldots\ \mathit{Sentence} \\
 & & \\
\mathit{Term} & \rightarrow & \mathit{Function}(\mathit{Term}, \ldots) \\
 & \mid & \mathit{Constant} \\
 & \mid & \mathit{Variable} \\
 & & \\
\mathit{Quantifier} & \rightarrow & \forall \mid \exists \\
\mathit{Constant} & \rightarrow & A \mid X_1 \mid \mathit{John} \mid \cdots \\
\mathit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\mathit{Predicate} & \rightarrow & \mathit{True} \mid \mathit{False} \mid \mathit{After} \mid \mathit{Loves} \mid \mathit{Raining} \mid \cdots \\
\mathit{Function} & \rightarrow & \mathit{Mother} \mid \mathit{LeftLeg} \mid \cdots \\
\textsc{Operator Precedence} & : & \neg, =, \land, \lor, \Rightarrow, \Leftrightarrow
\end{array}
$$

FOL concepts: Term and Variable

- A term is a logical expression that refers to an object.
    - E.g., John, LeftLeg(John), etc.

$$\text{Term} = \underbrace{\mathit{function}(\mathit{term}_1,\ldots,\mathit{term}_n)}_{\text{Complex term}} \text{ or } \mathit{constant} \text{ or } \mathit{variable}$$

- A complex term is formed by a function symbol followed by a parenthesized list of terms as arguments.
- A variable may serve as the argument of a function.
    - E.g., in predicates $King(x)$ or in function $LeftLeg(x)$.
- A term with no variable is called a ground term.

Terms and intended interpretation

- Consider a term, $f(t_1, .., t_n)$, that refers to some function F
- The arguments refer to objects in the domain, $d_1, .., d_n$.
- The whole term refers to the object resulting from applying F to $d_1, .., d_n$.
- For example,
  - Assume that $LeftLeg$ expresses the mapping between a person and his left leg, and $John$ stands for King John.
  - Then, $LeftLeg(John)$ refers to King John’s left leg.
- In this way, the interpretation fixes the referent of every term.

FOL concepts: Atomic sentence

- An atomic sentence states facts by using a predicate symbol followed by a parenthesized list of terms.
    - E.g., Brother(Richard, John), Married(Father(Richard), Mother(John))

$$\text{Atomic sentence} = \mathit{predicate}(\mathit{term}_1,\ldots,\mathit{term}_n)$$

- It is **true** if the relation referred to by the predicate symbol **holds** among the objects referred to by the arguments.

FOL concepts: Complex sentence

- We form a complex sentence from atomic ones by using logical connectives.
- For example,
  - $\neg\ Brother\ (LeftLeg(Richard), John)$
  - $Brother\ (Richard\ , John) \land Brother\ (John, Richard)$
  - $King(Richard) \lor King(John)$
  - $\neg\ King(Richard) \Rightarrow King(John)$
  - …
- Sentences are true relative to a model and an interpretation.

Quantifiers: Universal quantification

$\forall$***<variables> <sentence>***

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

- It is equivalent to the conjunction of instantiations of P.
- Typically, $\Rightarrow$ is the main connective with $\forall$
  - The conclusion is just for those objects for whom the premise is true
  - It says nothing at all about individuals for whom the premise is false.
- **Common mistake:** use $\wedge$ as the main connective with $\forall$ $\rightarrow$ too strong implication.
  - $\forall x\ Student(x, FIT) \wedge Smart(x)$ means "Everyone is a FIT student and Everyone is smart."

Quantifiers: Existential quantification

∃<variables> <sentence>

∃ x *P* is true in a model *m* iff *P* is true with *x* being some possible object in the model.

- It is equivalent to the disjunction of instantiations of P.
- Typically, ∧ as the main connective with ∃
- **Common mistake:** use ⇒ as the main connective with ∃ → too weak implication.
  - $\exists x\ Student(x, FIT) \Rightarrow Smart(x)$ true even with anyone not at FIT.

Quantifiers: Nested quantifiers

- Multiple quantifiers enable more complex sentences.
- The order of quantification is therefore very important.
    - $\forall x \exists y \ Loves(x, y)$ – “Everybody loves somebody.”
    - $\exists x \forall y \ Loves(y, x)$ – “There is someone loved by everyone.”

- Two quantifiers with the same variable name leads to confusion.

$$\forall x \ \left(Crown(x) \lor \left(\exists x \ Brother(Richard, x)\right)\right)$$

- A variable belongs to the innermost quantifier that mentions it.
- **Solution:** Use different variable names for nested quantifiers, e.g., $\exists z \ Brother(Richard, z)$

Quantifiers: Rules for duality

- ∀ and ∃ relate to each other through negation.

De Morgan's rules

$$\begin{aligned}
\forall x\ \neg P &\equiv \neg\exists x\ P & \neg(P \lor Q) &\equiv \neg P \land \neg Q \\
\neg\forall x\ P &\equiv \exists x\ \neg P & \neg(P \land Q) &\equiv \neg P \lor \neg Q \\
\forall x\ P &\equiv \neg\exists x\ \neg P & P \land Q &\equiv \neg(\neg P \lor \neg Q) \\
\exists x\ P &\equiv \neg\forall x\ \neg P & P \lor Q &\equiv \neg(\neg P \land \neg Q)
\end{aligned}$$

- For example,
  - $\forall x\ \ Likes(x, IceCream)$ $\quad\quad$ $\neg\exists x\ \ \neg Likes(x, IceCream)$
  - $\exists x\ \ Likes(x, Brocoli)$ $\quad\quad$ $\neg\forall x\ \ \neg Likes(x, IceCream)$

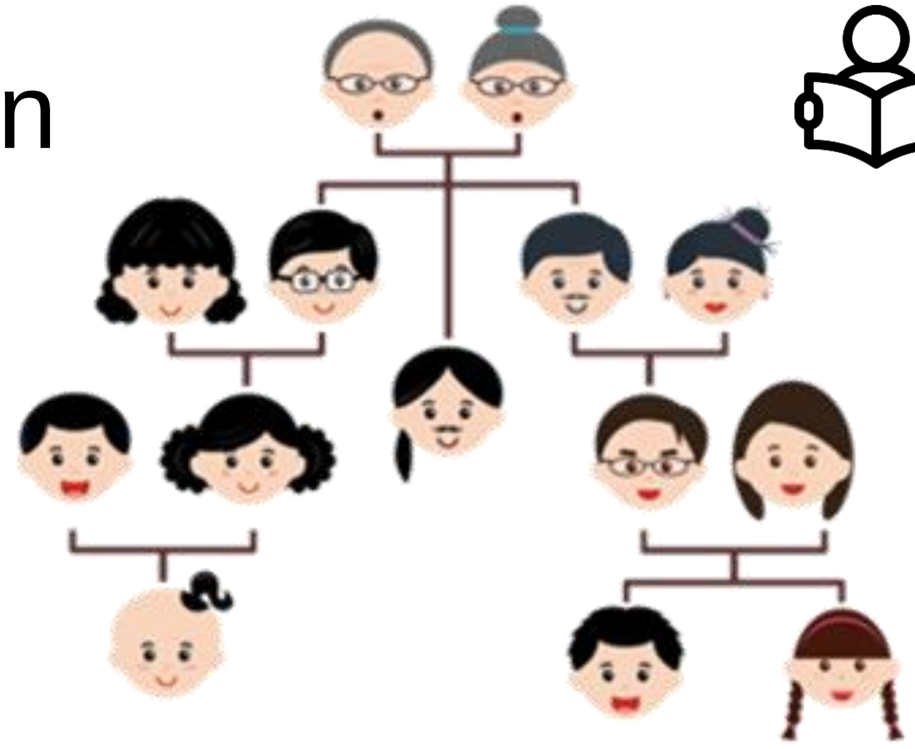
Equality symbol =

- $term_1 = term_2$ is true under a given interpretation iff $term_1$ and $term_2$ refer to the same object
  - E.g., $Father(John) = Henry$ means that $Father(John)$ and $Henry$ refer to the same object.
  - It states facts about a given function
- The negation insists that two terms are not the same.

  $\exists x, y \; Brother(x, Richard) \land Brother(y, Richard) \boxed{\land \neg(x = y)}$

Writing FOL sentences: Case studies

The kinship domain



- Unary predicates
  - Male and Female
- Binary predicates represent kinship relations.
  - Parenthood, brotherhood, marriage, etc.
  - Parent, Sibling, Brother , Sister, Child, Daughter, Son, Spouse, Wife, Husband, Grandparent , Grandchild , Cousin, Aunt, and Uncle.
- Functions
  - Mother and Father, each person has exactly one of each of these.

The kinship domain: Axioms

- One's mother is one's female parent

$$\forall m, c \quad Mother(c) = m \Leftrightarrow Female(m) \land Parent(m, c)$$

- One's husband is one's male spouse:

$$\forall w, h \;\; Husband(h, w) \Leftrightarrow Male(h) \land Spouse(h, w)$$

- Male and female are disjoint categories:

$$\forall x \;\; Male(x) \Leftrightarrow \neg Female(x)$$

- Parent and child are inverse relations:

$$\forall p, c \;\; Parent(p, c) \Leftrightarrow Child(c, p)$$

- A grandparent is a parent of one's parent:

$$\forall g, c \;\; Grandparent(g, c) \Leftrightarrow \exists p \;\; Parent(g, p) \land Parent(p, c)$$

- A sibling is another child of one's parents:

$$\forall x, y \;\; Sibling(x, y) \Leftrightarrow \neg(x = y) \land \exists p \;\; Parent(p, x) \;\land Parent(p, y)$$

The kinship domain: Theorems

- Theorems: logical sentences that are entailed by the axioms
  - E.g., $\forall x, y \;\; Sibling(x, y) \Leftrightarrow Sibling(y, x)$
- Theorems reduce the cost of deriving new sentences.
  - They do not increase the set of conclusions that follow from $KB \rightarrow$ no value from a pure logical point of view
  - They are essential from a practical point of view.

The set domain

- Sets are the empty set and those made by adjoining something to a set.
  - $\forall s\ Set(s) \Leftrightarrow (s = \{\,\}) \vee (\exists x, s_2\ Set(s_2) \wedge s = \{x|s_2\})$
- The empty set has no elements adjoined into it.
  - $\neg\exists x, s\ \ \{x|s\} = \{\,\}$
- Adjoining an element already in the set has no effect:
  - $\forall x, s\ \ x \in s \Leftrightarrow s = \{x|s\}$
- The only members of a set are the elements that were adjoined into it.
  - $\forall x, s\ x \in s \Leftrightarrow \exists y, s_2\ (s = \{y|s_2\} \wedge (x = y\ \vee\ x \in s_2))]$

The set domain

- Can you interpret the following sentences?
  - $\forall s_1, s_2 \;\; s_1 \sqsubseteq s_2 \;\Leftrightarrow\; (\forall x \;\; x \in s_1 \Rightarrow x \in s_2)$
  - $\forall s_1, s_2 \;\; s_1 = s_2 \;\Leftrightarrow\; (s_1 \sqsubseteq s_2 \wedge s_2 \sqsubseteq s_1)$
  - $\forall x, s_1, s_2 \;\; x \in (s_1 \cap s_2) \;\Leftrightarrow\; (x \in s_1 \wedge x \in s_2)$
  - $\forall x, s_1, s_2 \;\; x \in (s_1 \cup s_2) \;\Leftrightarrow\; (x \in s_1 \vee x \in s_2)$

Wumpus world: Input – Output

- Typical percept sentence
  - Percept([Stench, Breeze, Glitter, None, None]. 5)
- Actions
  - Turn(Right), Turn(Left), Forward, Shoot, Grab, Release, Climb
- To determine the best action, construct a query
  - $ASKVARS(\exists a\ BestAction(a, 5))$
  - Returns a **binding list** such as $\{a/Grab\}$

Wumpus world: The KB

- Perception
  - $\forall\ t, s, g, m, c\ Percept\ ([s, Breeze, g, m, c], t)\ \Rightarrow\ Breeze(t)$
  - $\forall\ t, s, b, m, c\ Percept\ ([s, b, Glitter, m, c], t)\ \Rightarrow\ Glitter\ (t)\quad \ldots$
- Reflex
  - $\forall t\ Glitter(t)\ \Rightarrow\ BestAction(Grab, t)$
- Environment definition:

$$\forall x, y, a, b\quad Adjacent([x, y], [a, b]) \Leftrightarrow$$
$$\big(x = a\ \wedge\ (y = b - 1\ \vee\ y = b\ +\ 1)\big)$$
$$\vee\ (y = b\ \wedge\ (x = a - 1\ \vee\ x = a\ +\ 1))$$

Wumpus world: Hidden properties

- Properties of squares

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

- Squares are breezy near a pit
  - Diagnostic rule --- infer cause from effect

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

  - Causal rule --- infer effect from cause

$$\forall r\ Pit(r) \Leftrightarrow [\forall s\ \ Adjacent(r,s) \Rightarrow Breezy(s)]$$

Quiz 01: Writing FOL sentences

- Given the following predicates

  Pet(x): x is a pet   Dog(x): x is a dog   Cat(x): x is a cat

  Larger(x, y): x is larger than y

- For each English sentence below, write the FOL sentence that best expresses its intended meaning using only the given predicates.
  1. If Bingo is a dog then Bingo is not a cat.
  2. Every pet is either a dog or a cat.
  3. No dog is a cat.
  4. Some dog is larger than every cat.
  5. Every dog is larger than some cat.
  6. Bingo is a dog who is larger than at least two cats.

First-order inference

$\forall x. Cat(x) \Rightarrow Cute(x)$

Universal / Existential Instantiation

- Let $SUBST(\theta, \alpha)$ be the result of applying the substitution θ to the sentence α.
- Universal Instantiation (UI) rule:
  - for any variable v and ground term g.

$$\frac{\forall \boldsymbol{v} \ \ \boldsymbol{\alpha}}{\textbf{SUBST}(\{\boldsymbol{v}/\boldsymbol{g}\}, \boldsymbol{\alpha})}$$

- Existential Instantiation (EI) rule:

$$\frac{\exists \boldsymbol{v} \ \ \boldsymbol{\alpha}}{\textbf{SUBST}(\{\boldsymbol{v}/\boldsymbol{k}\}, \boldsymbol{\alpha})}$$

  - for any sentence α, variable v, and constant symbol k
  - k must not appear elsewhere in KB, called **Skolem constant**.

Universal / Existential Instantiation

- We can infer from $\forall \boldsymbol{x}\ \boldsymbol{King}(\boldsymbol{x}) \land \boldsymbol{Greedy}(\boldsymbol{x}) \Rightarrow \boldsymbol{Evil}(\boldsymbol{x})$ any of the below.
    - $King(John) \land Greedy(John) \Rightarrow Evil(John)$
    - $King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)$
    - $King(Father(John)) \land Greedy\big(Father(John)\big) \Rightarrow Evil(Father(John))$
    - ...
    - with substitutions $\{x/John\}$, $\{x/Richard\}$, and $\{x/Father(John)\}$

- However, from $\exists x\ \ Crown(x) \land OnHead(x, John)$, we only infer $Crown(C_1) \land\ OnHead(C_1, John)$ once.

Universal / Existential Instantiation

- The **UI** rule can be applied many times to produce different consequences.
- The **EI** rule can be applied once, and then the existentially quantified sentence is discarded.
  - E.g., discard $\exists x\ Kill(x, Victim)$ after adding $Kill(Murderer, Victim)$
  - The new KB is not **logically equivalent** to the old but shown to be **inferentially equivalent**.

Reduction to propositional inference

- Frst-order inference can be reduced to propositional one.
- The first-order KB is essentially propositional if we view the ground atomic sentences as proposition symbols.
- A ground sentence is entailed by new KB iff it is entailed by the original KB.

Reduction to propositional inference

- For example, suppose the KB contains just the sentences

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

  $King(John)$

  $Greedy(John)$

  $Brother(Richard, John)$

- Apply UI with substitutions, $\{x/John\}$ and $\{x/Richard\}$

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

  $King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)$

Reduction to propositional inference

- **Problem:** *When the KB includes a function symbol, the set of possible ground-term substitutions is infinite.*
  - E.g., the $Father$ symbol, infinitely many nested terms such as $Father(Father(Father(John)))$ can be constructed.

- **Herbrand's Theorem (1930):** *If an original FOL $KB \vDash \alpha$, α is entailed by a **finite** subset of the propositionalized KB.*
  - For $n = 0\ to\ \infty$ do
    - Create a propositional KB by instantiating with depth-*n* terms
    - See if α is entailed by this KB
  - $n = 0$: $Richard$ and $John$
  - $n = 1$: $Father(Richard)$ and $Father(John)$, etc.

Reduction to propositional inference

- **Problem:** *The inference works if sentence α is entailed, but it loops if α is not entailed.*
- **Theorems of Turing (1936) and Church (1936):** *The question of entailment for first-order logic is semidecidable.*
  - Algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every non-entailed sentence.

Unification and Lifting

Unification

- Make different logical expressions identical with substitutions

$$\boldsymbol{UNIFY(p,q) = \theta} \text{ where } SUBST(\theta,p) = SUBST(\theta,q)$$

- For example,

p	***q***	***θ***
Knows(John, *x*)	Knows(John, Jane)	{*x*/Jane}
Knows(John, *x*)	Knows(*y*, Steve)	{*x*/Steve, *y*/John}
Knows(John, *x*)	Knows(*y*, Mother(*y*))	{*x*/Mother(John), *y*/John}
Knows(John, *x*)	Knows(*x*, Steve)	*fail*

*Problem is due to use of same variable **x** in both sentences*

Standardizing apart eliminates overlap of variables Knows(*z*, Steve)

Most General Unifier (MGU)

- $\text{UNIFY}(Knows(John, x), Knows(y, z)) = \theta$
    1. $\theta = \{y/John, x/z\}$
    2. $\theta = \{y/John, x/John, z/John\}$
- 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.

$$MGU = \{y/John, x/z\}$$

MGU: Some examples

$\boldsymbol{\omega_1}$	$\boldsymbol{\omega_2}$	$\boldsymbol{MGU}$
$A(B, C)$	$A(x, y)$	$\{x/B, y/C)$
$A(x, f(D, x))$	$A(E, f(D, y))$	$\{x/E, y/E\}$
$A(x, y)$	$A(f(C, y), z)$	$\{x/f(C, y), y/z\}$
$P(A, x, f(g(y)))$	$P(y, f(z), f(z))$	$\{y/A, x/f(z), z/g(A)\}$
$P(x, g(f(A)), f(x))$	$P(f(y), z, y)$	No MGU
$P(x, f(y))$	$P(z, g(w))$	No MGU

Unification algorithm

```
function UNIFY($x$ , $y$, $\theta$) returns a substitution to make $x$ and $y$ identical
    inputs: $x$ , a variable, constant, list, or compound expression
            $y$, a variable, constant, list, or compound expression
            $\theta$, the substitution built up so far (optional, defaults to empty)
    if $\theta$ = failure then return failure
    else if $x = y$ then return $\theta$
    else if VARIABLE?($x$) then return UNIFY-VAR($x$ , $y$, $\theta$)
    else if VARIABLE?($y$) then return UNIFY-VAR($y$, $x$ , $\theta$)
    else if COMPOUND?($x$) and COMPOUND?($y$) then
        return UNIFY($x$.ARGS, y.ARGS, UNIFY($x$.OP, $y$.OP, $\theta$))
    else if LIST?($x$) and LIST?($y$) then
        return UNIFY($x$.REST, $y$.REST, UNIFY($x$.FIRST, $y$.FIRST, $\theta$))
    else return failure
```

Unification algorithm

```
function UNIFY-VAR($var, x, \theta$) returns a substitution
    if $\{var/val\} \in \theta$ then return UNIFY($val, x, \theta$)
    else if $\{x/val\} \in \theta$ then return UNIFY($var, val, \theta$)
    else if OCCUR-CHECK?($var, x$) then return failure
    else return add $\{var/x\}$ to $\theta$
```

a is a constant

$$\begin{tikzcd}
\text{Atom\_mgu}(\ P(a,x,f(g(y))),\ P(z,f(z),f(w))) \arrow[d] & \\
\text{Term\_list\_mgu}(\ \langle a,\ x,\ f(g(y))\rangle,\ \langle z,\ f(z),\ f(w)\rangle,\ \varnothing) \arrow[d] & \\
\text{Term\_list\_mgu\_aux}(\ \langle x,\ f(g(y))\rangle,\ \langle f(z),\ f(w)\rangle,\ \text{Term\_mgu}(a,\ z,\ \varnothing),\ \varnothing) \arrow[dd] \arrow[dr] & \\
 & \text{Variable\_mgu}(z,\ a,\ \varnothing) \arrow[d] \\
\text{Term\_list\_mgu}(\ \langle x,\ f(g(y))\rangle,\ \langle f(a),\ f(w)\rangle,\ \{a/z\} \arrow[d] & \{a/z\} \\
\text{Term\_list\_mgu\_aux}(\ \langle f(g(y))\rangle,\ \langle f(w)\rangle,\ \text{Term\_mgu}(x,\ f(a),\ \varnothing),\ \{a/z\}) \arrow[ddd] \arrow[dr] & \\
 & \text{Variable\_mgu}(x,\ f(a),\ \varnothing) \arrow[d] \\
 & \{f(a)/x\} \\
{} &
\end{tikzcd}$$

$$\begin{tikzcd}
\text{Term\_list\_mgu}(\langle\rangle, \langle\rangle, \text{Term\_mgu}(f(g(y)), f(w), \{a/z, f(a)/x\})) \arrow[d] & \\
\text{Term\_list\_mgu}(\langle g(y)\rangle, \langle w\rangle, \{a/z, f(a)/x\})) \arrow[d] & \\
\text{Term\_list\_mgu\_aux}(\langle\rangle, \langle\rangle, \text{Term\_mgu}(g(y), w, \varnothing) \{a/z, f(a)/x\})) \arrow[dddd] \arrow[dr] & \\
& \text{Variable\_mgu}(w, g(y), \varnothing) \arrow[d] \\
& \{ g(y)/w\} \\
& \\
\text{Term\_list\_mgu}(\langle\rangle, \langle\rangle, \{a/z, f(a)/x, g(y)/w\})) \arrow[d] & \\
\boxed{\{a/z, f(a)/x, g(y)/w\}} &
\end{tikzcd}$$

Quiz 02: Find the MGU

- Find the MGU when performing $UNIFY(p, q)$

$\boldsymbol{\omega_1}$	$\boldsymbol{\omega_2}$	$\boldsymbol{MGU}$
$P(f(A), g(x))$	$P(y, y)$	?
$P(A, x, h(g(z)))$	$P(z, h(y), h(y))$	?
$P(x, f(x), z)$	$P(g(y), f(g(B)), y)$	?
$P(x, f(x))$	$P(f(y), y)$	?
$P(x, f(z))$	$P(f(y), y)$	?

Forward chaining

First-order inference rule

- **If** there is some substitution θ making each of the conjuncts of the premise identical to sentences already in the KB
- **Then** the conclusion can be asserted after applying θ.
- For example, consider the following KB

$$\forall x \;\; King(x) \land Greedy(x) \Rightarrow Evil(x)$$

$$\forall y \;\; Greedy(y)$$

$$King(John)$$

$$Brother(Richard, John)$$

  - Apply the substitutions $\{x/John, y/John\}$ to $King(x)$ and $Greedy(x)$, $King(John)$ and $Greedy(y)$, then they will be identical in pairs.
  - Thus, infer the conclusion of the implication

Generalized Modus Ponens (GMP)

- For atomic sentences p_i, p_i' and q, where there exists θ such that $SUBST(\theta, p_i') = SUBST(\theta, p_i)$, for all i,

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

- For 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 $Evil(x)$

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

- All variables assumed universally quantified
- A **lifted version** of Modus Ponens $\rightarrow$ **sound** inference rule

First-order definite clause

- A definite clause is a disjunction of literals of which exactly one is positive.
- It is either atomic or an implication with a positive consequent and a conjunctive antecedent.
  - E.g., $King(x) \wedge Greedy(x) \Rightarrow Evil(x), King(John), Greedy(y)$
- Variables in first-order literals are universally quantified.
- Not every KB can be converted into definite clauses due to the single-positive-literal restriction.

FOL definite clause: An example

- Consider the following problem

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.

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

$$\exists\, x\ Owns(Nono,x) \land Missile(x) \qquad \begin{cases} Owns(Nono, M_1) \\ Missile(M_1) \end{cases}$$

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

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

$$American(West)$$

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

$$Enemy(Nono, America)$$

```
function FOL-FC-ASK($KB$,$\alpha$) returns a substitution or false
  inputs: $KB$, the knowledge base, a set of first-order definite clauses
          $\alpha$, the query, an atomic sentence
  local variables: new, the new sentences inferred on each iteration
  repeat until new is empty
    $new \leftarrow \{\,\}$
    for each rule in KB do
      $(p_1 \wedge \cdots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-VARIABLES(rule)
      for each $\theta$ such that SUBST($\theta$, $p_1 \wedge \cdots \wedge p_n$) = SUBST($\theta$,$p'_1 \wedge \cdots \wedge p'_n$ )
                                  for some $p'_1, \ldots, p'_n$ in KB
        $q' \leftarrow$ SUBST($\theta$,$q$)
        if $q'$ does not unify with some sentence already in KB or new then
          add $q'$ to new
          $\varphi \leftarrow$ UNIFY($q'$,$\alpha$)
          if $\varphi$ is not fail then return $\varphi$
    add new to KB
  return false
```

Forward chaining: An example

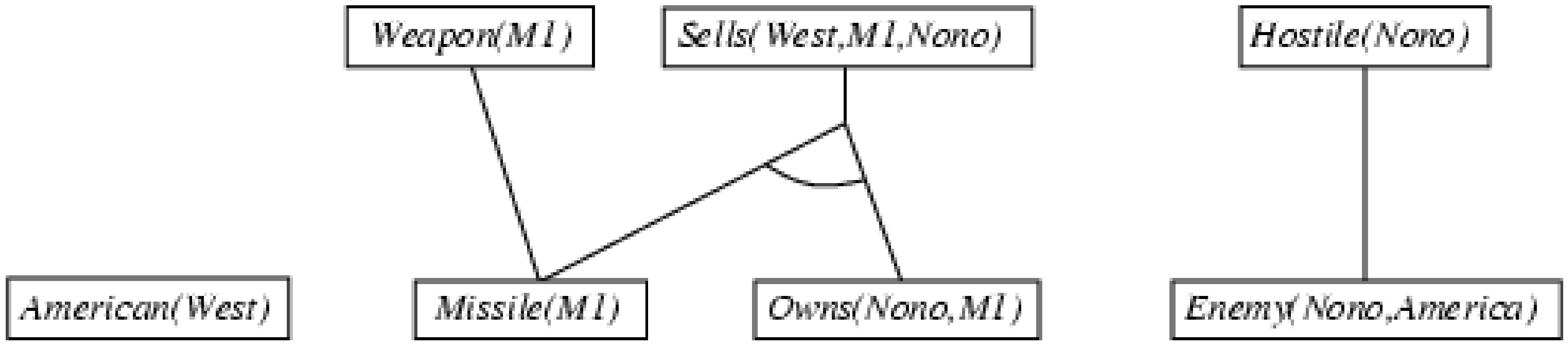
American(West) *Missile(M1)* *Owns(Nono,M1)* *Enemy(Nono,America)*

Forward chaining: An example

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

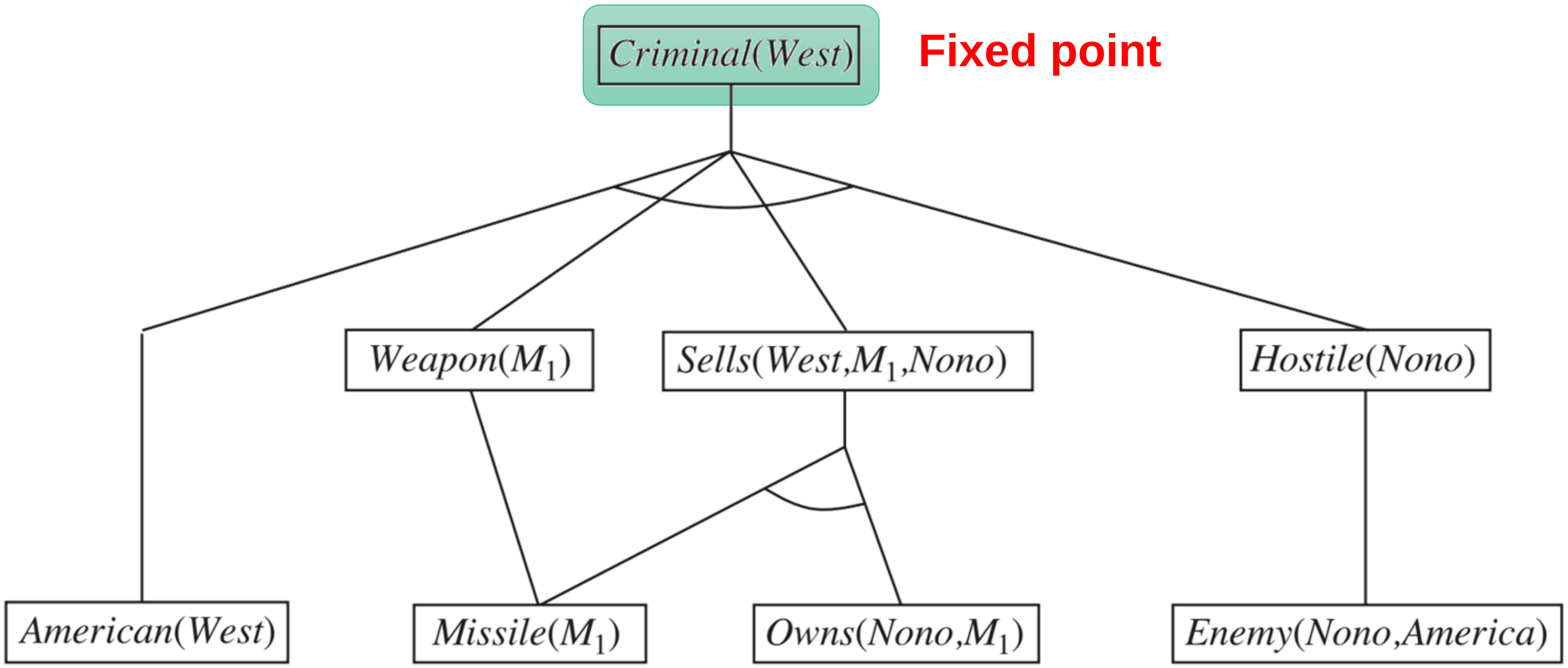
$Missile(x) \Rightarrow Weapon(x)$

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



Forward chaining: An example

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



Forward chaining

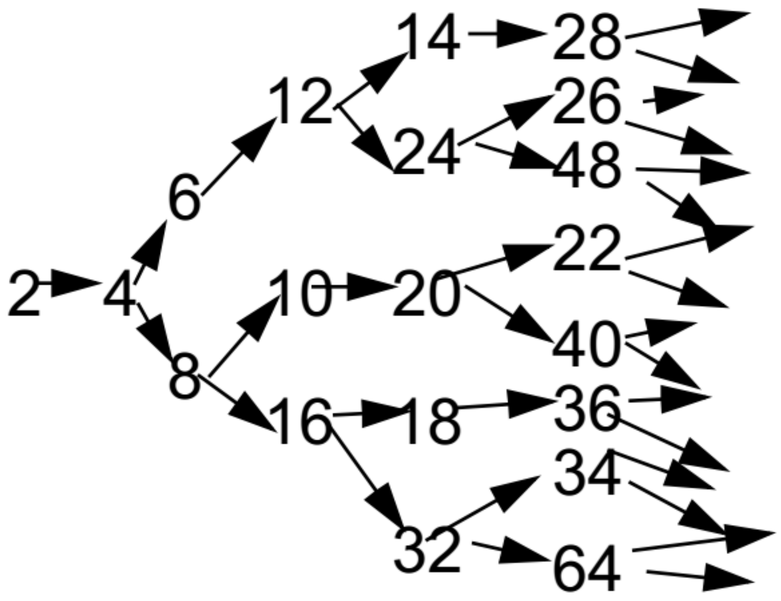
- Soundness?
  - YES, every inference is just an application of GMP.
- Completeness?
  - YES for definite clause knowledge bases.
  - It answers every query whose answers are entailed by any KB of definite clauses.
- Terminate for Datalog in finite number of iterations
  - **Datalog** = first-order definite clauses + no functions
- Entailment with definite clauses is **semidecidable**.
  - May not terminate in general if α is not entailed, unavoidable.

Renaming

- A fact is not **“new”** if it is just a renaming of a known fact.
- One sentence is a renaming of another if they are identical except for the names of the variables.
    - E.g., Likes(x, IceCream) vs. Likes(y, IceCream)

Definite clauses with function symbols

- Inference can explode forward and may never terminate.



$Even(x) \Rightarrow Even(plus(x, 2))$

$Integer(x) \Rightarrow Even(times(2, x))$

$Even(x) \Rightarrow Integer(x)$

$Even(2)$

Efficient forward chaining

- Incremental forward chaining
  - No need to match a rule on iteration k if a premise was not added on iteration $k-1$ $\rightarrow$ match each rule whose premise contains a newly added positive literal
- Matching itself can be expensive
- Database indexing allows $O(1)$ retrieval of known facts
  - E.g., query $Missile(x)$ retrieves $Missile(M_1)$
- Widely used in deductive databases

Quiz 03: Forward chaining

- Given a KB containing the following sentence
  1. $Parent(x, y) \wedge Male(x) \Rightarrow Father(x, y)$
  2. $Father(x, y) \wedge Father(x, z) \Rightarrow Sibling(y, z)$
  3. $Parent(Tom, John)$
  4. $Male(Tom)$
  5. $Parent(Tom, Fred)$
- Perform the forward chaining until a fixed point is reached.

Backward chaining

```
function FOL-BC-ASK(KB,query) returns a generator of substitutions
    return FOL-BC-OR(KB,query, { })
```

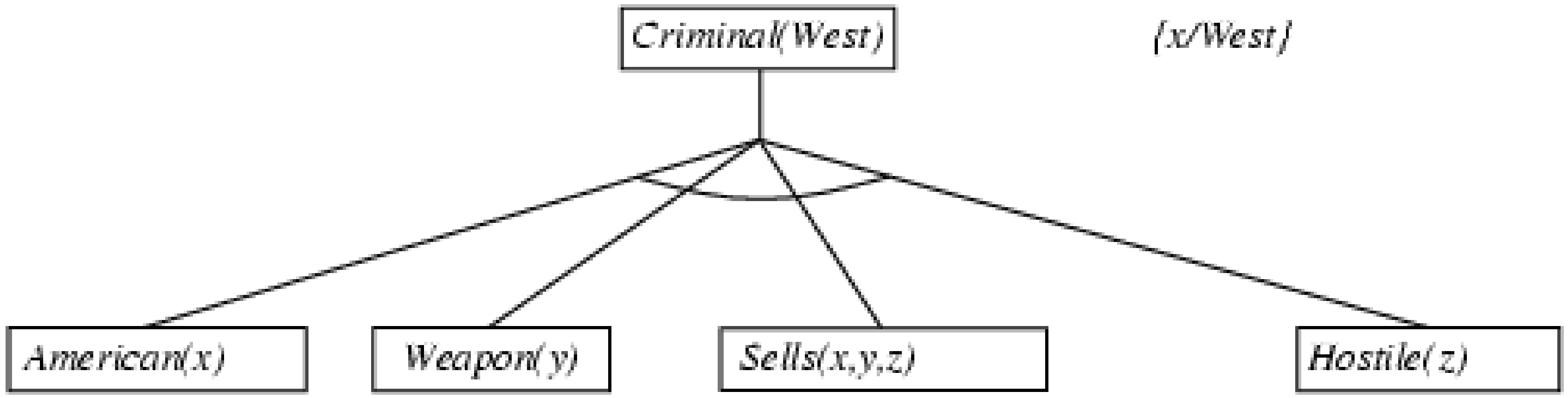
```
generator FOL-BC-OR(KB,goal, $\theta$) yields a substitution
    for each rule (lhs $\Rightarrow$ rhs) in FETCH-RULES-FOR-GOAL(KB, goal) do
        (lhs, rhs) $\leftarrow$ STANDARDIZE-VARIABLES((lhs, rhs))
        for each $\theta'$ in FOL-BC-AND(KB,lhs, UNIFY(rhs, goal, $\theta$)) do
            yield $\theta'$
```

```
generator FOL-BC-AND(KB,goals, $\theta$) yields a substitution
    if $\theta$ = failure then return
    else if LENGTH(goals) = 0 then yield $\theta$
    else do
        first,rest $\leftarrow$ FIRST(goals), REST(goals)
        for each $\theta'$ in F'OL-BC-OR(KB, SUBST($\theta$, first), $\theta$) do
            for each $\theta''$ in FOL-BC-AND(KB,rest, $\theta'$) do
                yield $\theta''$
```

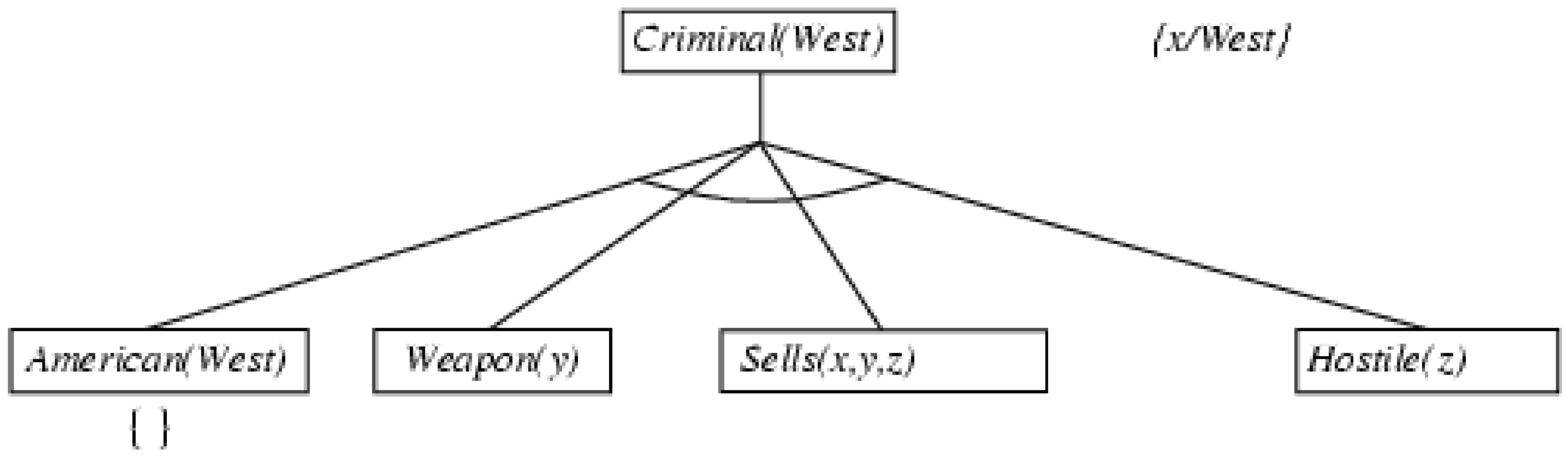
Backward chaining: An example

Criminal(West)

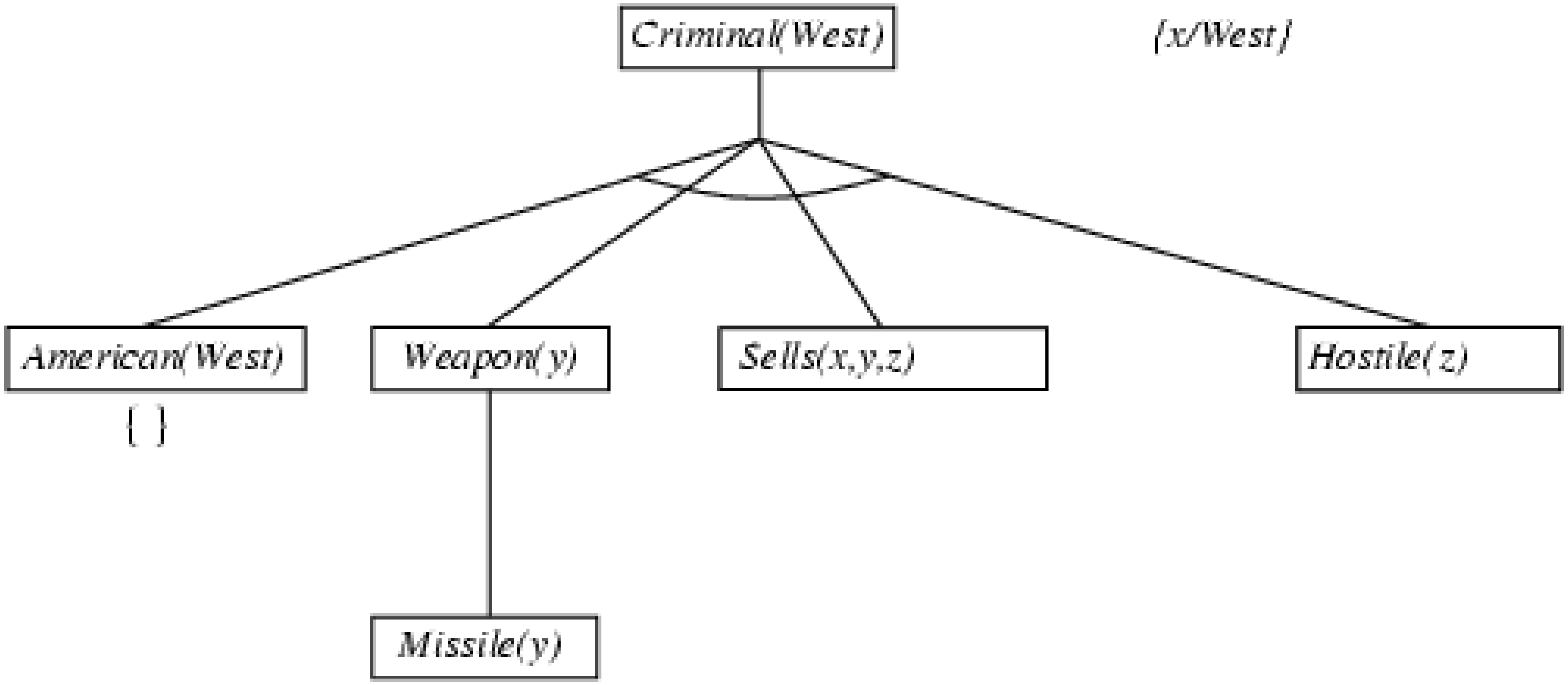
Backward chaining: An example



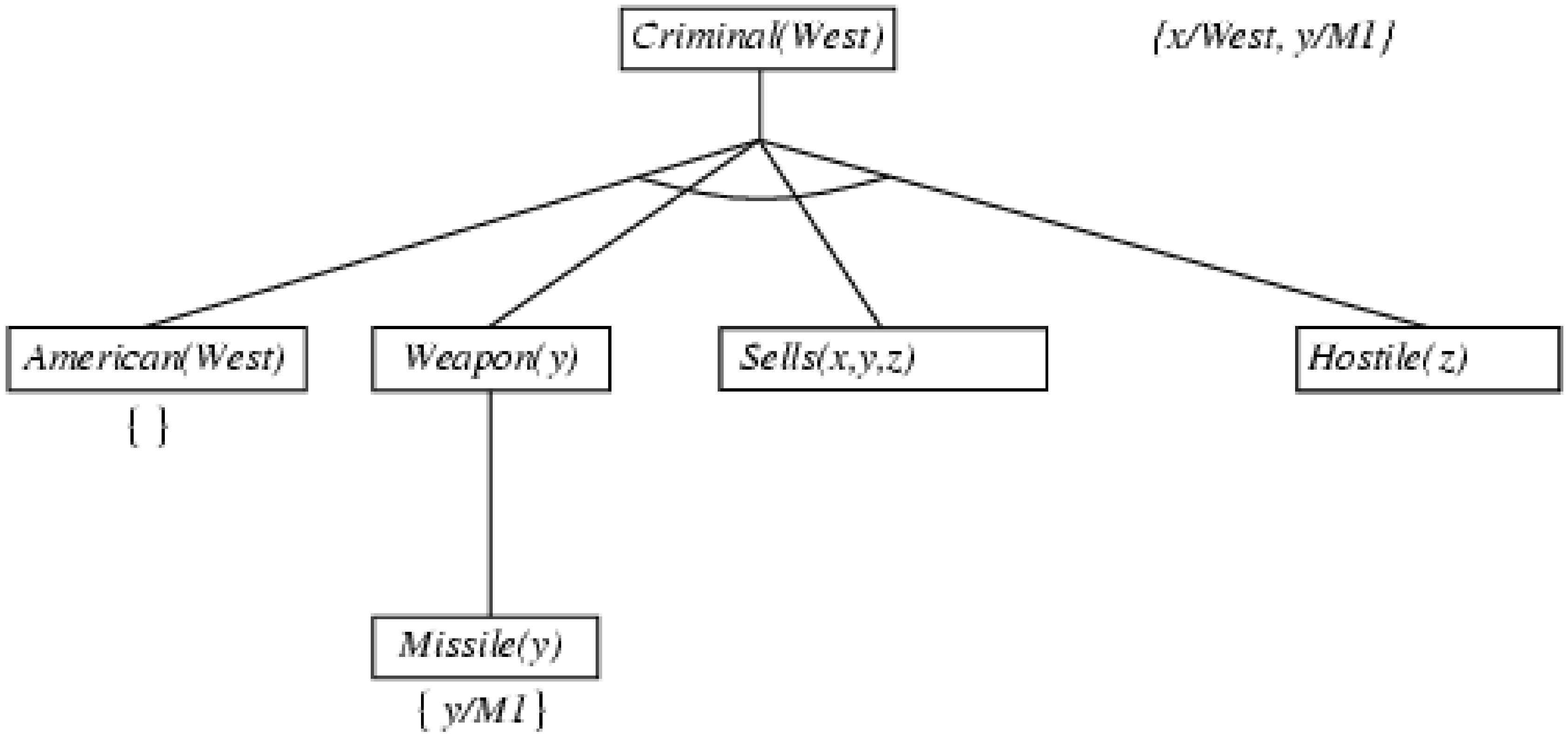
Backward chaining: An example



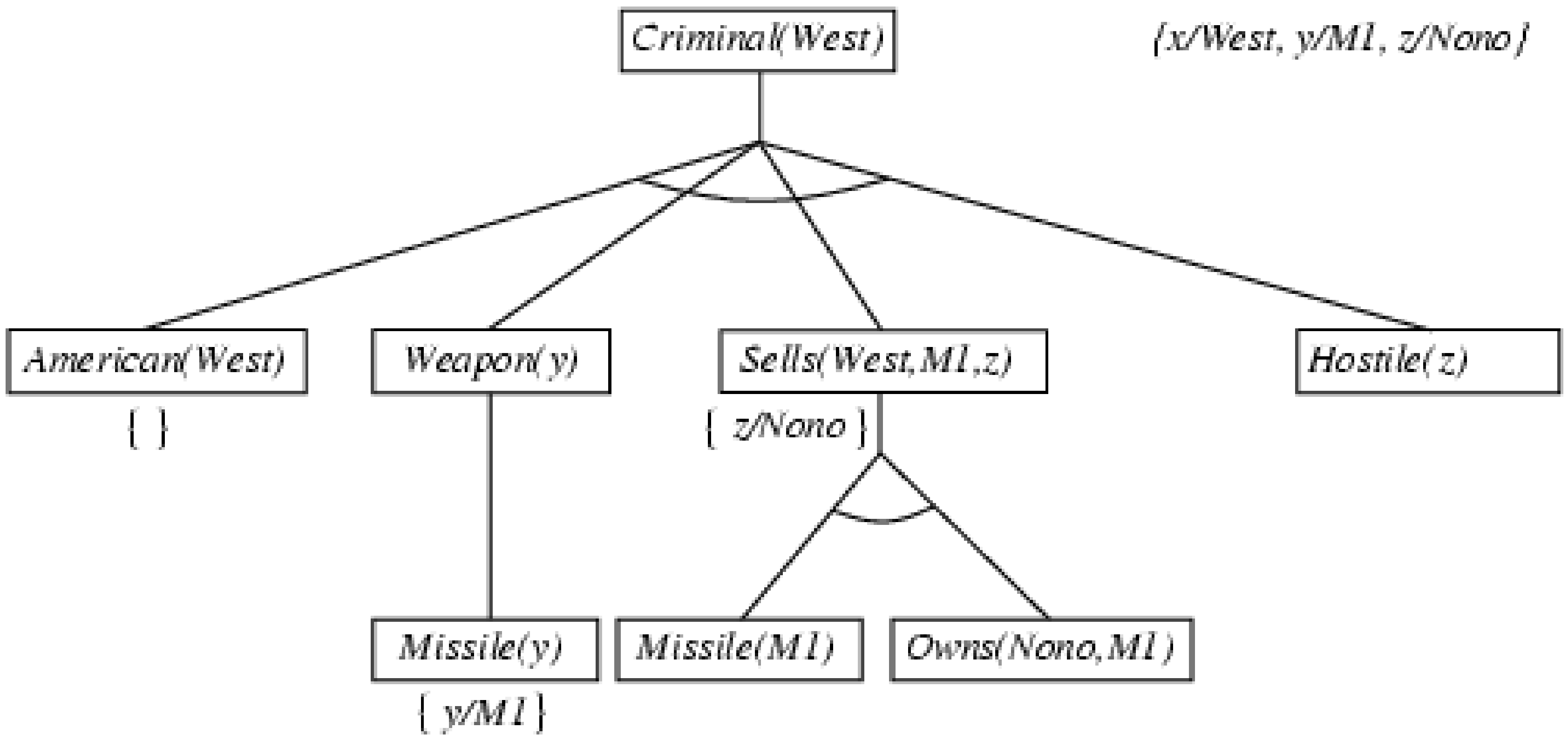
Backward chaining: An example



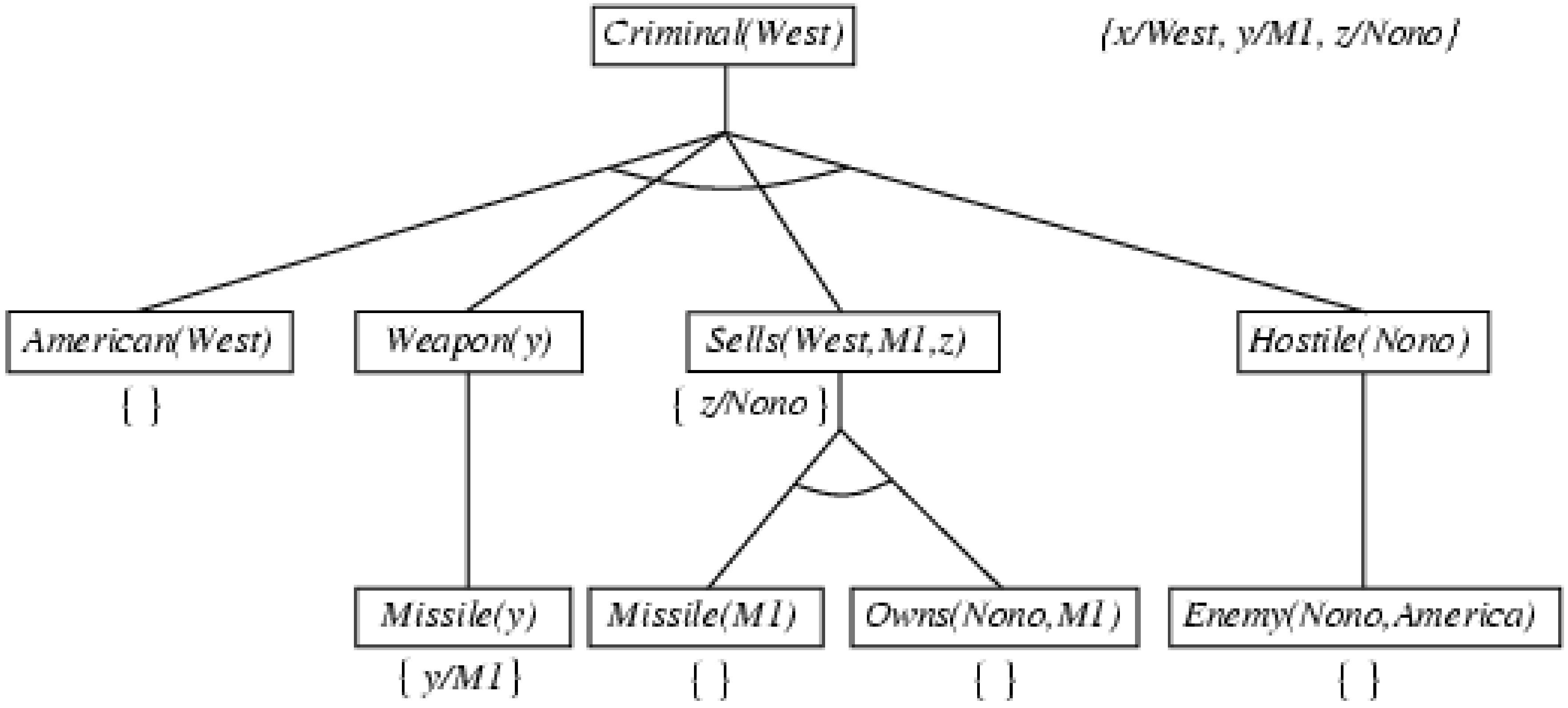
Backward chaining: An example



Backward chaining: An example



Backward chaining: An example



Properties of backward chaining

- Depth-first recursive proof search
  - Space is linear in size of proof
- Incomplete due to infinite loops
  - Fix by checking current goal against every goal on stack
- Inefficient due to repeated subgoals (success and failure)
  - Fix using caching of previous results (extra space)
- Widely used for logic programming

Quiz 04: Backward chaining

- Given a KB containing the following sentence
  1. $Parent(x, y) \wedge Male(x) \Rightarrow Father(x, y)$
  2. $Father(x, y) \wedge Father(x, z) \Rightarrow Sibling(y, z)$
  3. $Parent(Tom, John)$
  4. $Male(Tom)$
  5. $Parent(Tom, Fred)$
- Find solution(s) to each of the following queries
  - Parent(Tom, x)
  - Father(Tom, s)
  - Father(f, s)
  - Sibling(a, b)

Quiz 04: Backward chaining

- Query: Parent(Tom,x)
  - Answers: ({x/John}, {x/Fred})
- Query: Father(Tom,s)
  - Subgoal: Parent(Tom,s) ∧ Male(Tom)
    - {s/John}
      - Subgoal: Male(Tom)
    - Answer: {s/John}
    - {s/Fred}
      - Subgoal: Male(Tom)
    - Answer: {s/Fred}
  - Answers: ({s/John}, {s/Fred})

Resolution refutation

CNF for First-order logic

- First-order resolution requires that sentences be in CNF.
- For example, the sentence

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

becomes, in CNF,

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

- *Every sentence of first-order logic can be converted into an inferentially equivalent CNF sentence.*
  - The CNF sentence will be unsatisfiable just when the original sentence is unsatisfiable $\rightarrow$ perform proofs by contraction.

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 implications

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

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

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

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

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

Conversion to CNF

Everyone who loves all animals is loved by someone.

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

3. Standardize variables: each quantifier uses a different one

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

4. Skolemize: remove existential quantifiers by elimination
   - Simple case: translate $\exists x\ P(x)$ into $P(A)$, where A is a new constant.
     - However, $\forall x\ [Animal(A) \wedge \neg Loves(x,A)] \vee [Loves(B,x)]$ has an entirely different meaning.
   - The arguments of the **Skolem function** are all universally quantified variables in whose scope the existential quantifier appears.

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

Conversion to CNF

Everyone who loves all animals is loved by someone.

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

5. Drop universal quantifiers

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

6. Distribute ∨ over ∧

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

Resolution inference rule

- Simply a lifted version of the propositional resolution rule
- Formulation

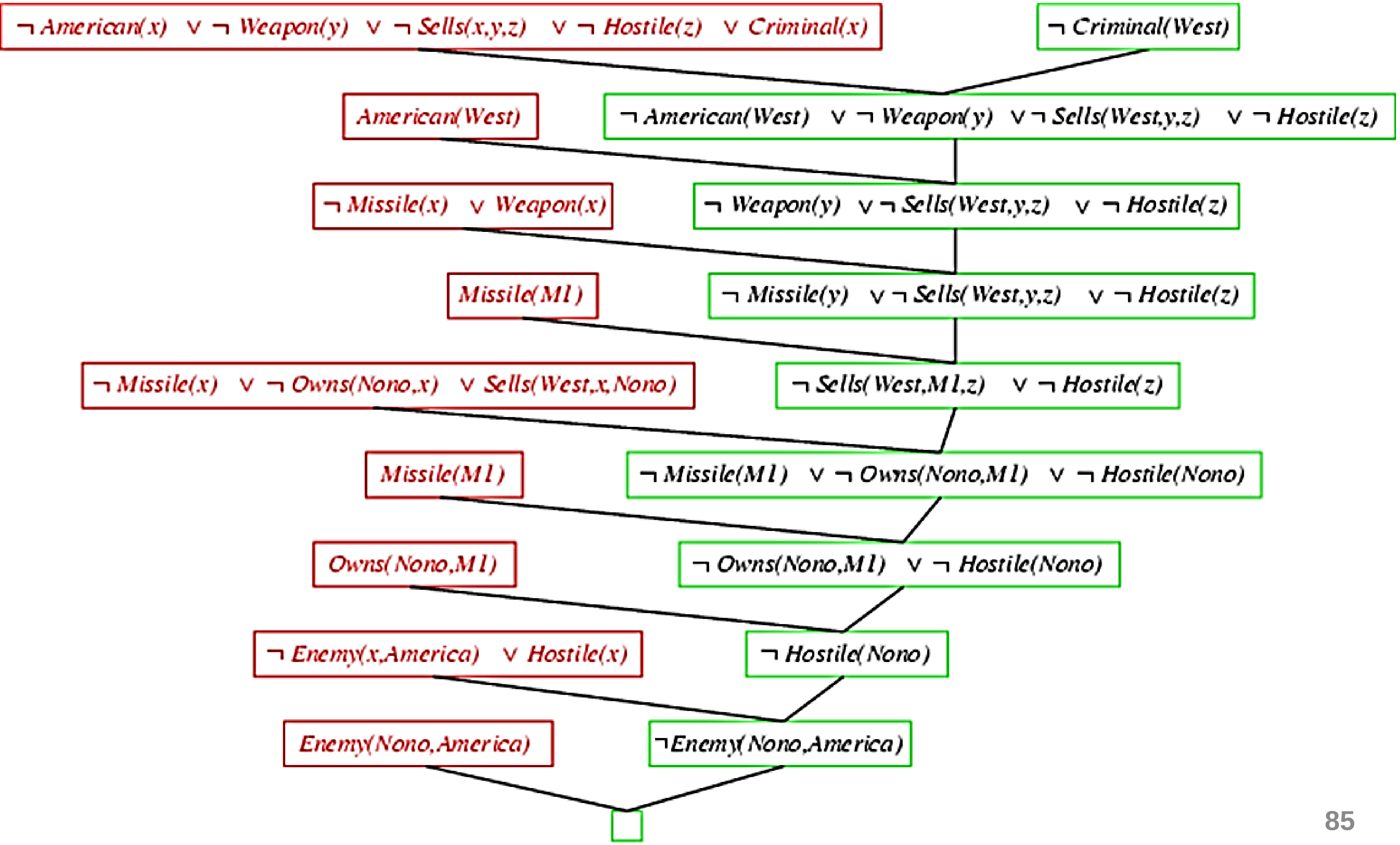
$$\frac{l_1 \vee \cdots \vee l_k \qquad m_1 \vee \cdots \vee m_n}{\mathbf{SUBST}(\boldsymbol{\theta}, l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

  - where $UNIFY(l_i, \neg m_j) = \theta$

- For example,

$$\frac{Animal(F(x)) \vee Loves(G(x), x) \qquad \neg Loves(u, v) \vee \neg Kills(u, v)}{Animal(F(x)) \vee \neg Kills(G(x), x)} \qquad \theta = \{u/G(x), v/x\}$$

Resolution: An example



¬American(x) ∨ ¬Weapon(y) ∨ ¬Sells(x,y,z) ∨ ¬Hostile(z) ∨ Criminal(x)
¬Criminal(West)
American(West)
¬American(West) ∨ ¬Weapon(y) ∨ ¬Sells(West,y,z) ∨ ¬Hostile(z)
¬Missile(x) ∨ Weapon(x)
¬Weapon(y) ∨ ¬Sells(West,y,z) ∨ ¬Hostile(z)
Missile(M1)
¬Missile(y) ∨ ¬Sells(West,y,z) ∨ ¬Hostile(z)
¬Missile(x) ∨ ¬Owns(Nono,x) ∨ Sells(West,x,Nono)
¬Sells(West,M1,z) ∨ ¬Hostile(z)
Missile(M1)
¬Missile(M1) ∨ ¬Owns(Nono,M1) ∨ ¬Hostile(Nono)
Owns(Nono,M1)
¬Owns(Nono,M1) ∨ ¬Hostile(Nono)
¬Enemy(x,America) ∨ Hostile(x)
¬Hostile(Nono)
Enemy(Nono,America)
¬Enemy(Nono,America)

Resolution: Another example

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

C. $\forall x\ Animal(x) \Rightarrow Loves(Jack,x)$

D. $Kills(Jack,Tuna) \vee Kills(Curiosity,Tuna)$

E. $Cat(Tuna)$

F. $\forall x\ Cat(x) \Rightarrow Animal(x)$

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

Resolution: Another example

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. $Animal(F(x)) \vee Loves(G(x), x)$

$\neg Loves(x, F(x)) \vee Loves(G(x), x)$

B. $\neg Loves(y, x) \vee \neg Animal(z) \vee \neg Kills(x, z)$

C. $\neg Animal(x) \vee Loves(Jack, x)$

D. $Kills(Jack, Tuna) \vee Kills(Curiousity, Tuna)$

E. $Cat(Tuna)$

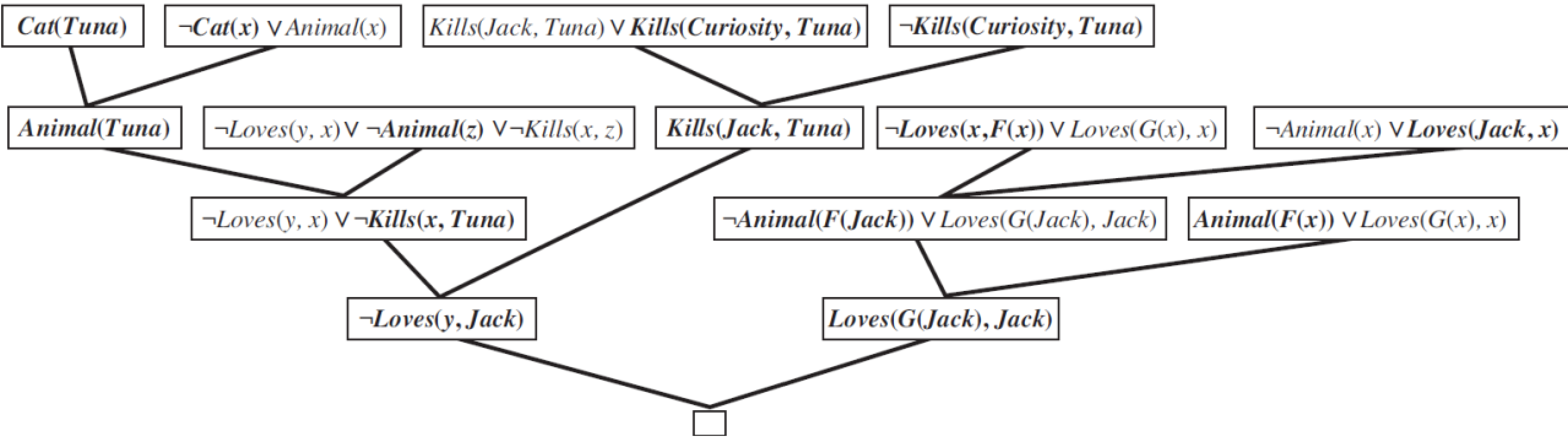
F. $\neg Cat(x) \vee Animal(x)$

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

Resolution: Another example

Suppose Curiosity did not kill Tuna. We know that either Jack or Curiosity did; thus Jack must have. Now, Tuna is a cat and cats are animals, so Tuna is an animal. Because anyone who kills an animal is loved by no one, we know that no one loves Jack. On the other hand, Jack loves all animals, so someone loves him; so we have a contradiction.

Therefore, Curiosity killed the cat.



Cat(Tuna)
¬Cat(x) ∨ Animal(x)
Kills(Jack, Tuna) ∨ Kills(Curiosity, Tuna)
¬Kills(Curiosity, Tuna)
Animal(Tuna)
¬Loves(y, x) ∨ ¬Animal(z) ∨ ¬Kills(x, z)
Kills(Jack, Tuna)
¬Loves(x, F(x)) ∨ Loves(G(x), x)
¬Animal(x) ∨ Loves(Jack, x)
¬Loves(y, x) ∨ ¬Kills(x, Tuna)
¬Animal(F(Jack)) ∨ Loves(G(Jack), Jack)
Animal(F(x)) ∨ Loves(G(x), x)
¬Loves(y, Jack)
Loves(G(Jack), Jack)

Quiz 05: Resolution

- Given a KB of the following sentences
  - Anyone whom Mary loves is a football star.
  - Any student who does not pass does not play.
  - John is a student.
  - Any student who does not study does not pass.
  - Anyone who does not play is not a football star.
- Prove that If John does not study, Mary does not love John.
- Write the FOL sentences using only the given predicates

Loves(x, y): “x loves y”	Star(x): “x is a football star”
Student(x): “x is a student”	Pass(x): “x passes”
Play(x): “x plays”	Study(x): “x studies”

...the end.