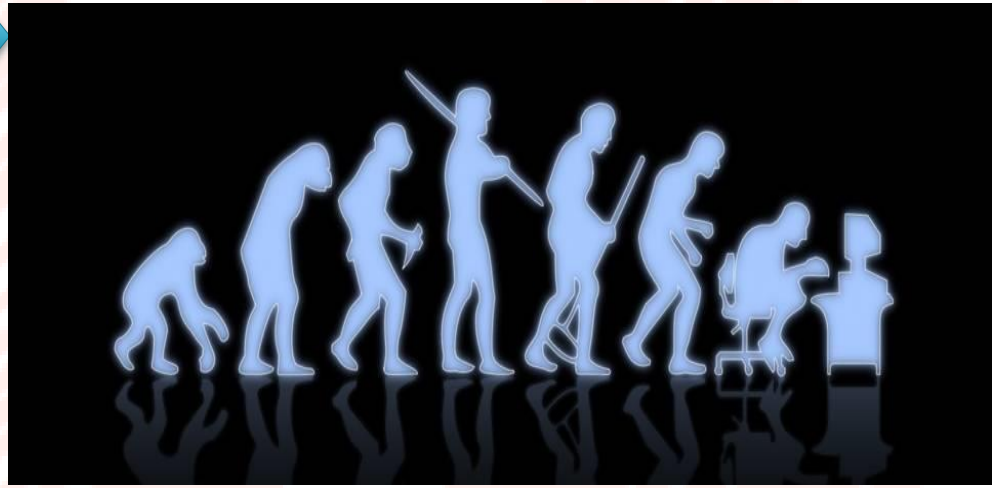
FACULTY OF INFORMATION TECHNOLOGY

Artificial Intelligence Fundamentals (NM TTNT)

Semester 1, 2021/2022

Chapter 7. First Order Logic



Content

- Why FOL?
- Syntax and semantics of FOL
- Using FOL
- Wumpus world in FOL
- Knowledge engineering in FOL

First-Order Logic

- Logics are characterized by what they consider to be "primitives"

Logic	Primitives	Available Knowledge
Propositional	facts (propositions)	true/false/unknown
First-Order	facts, objects, relations	true/false/unknown
Temporal	facts, objects, relations, times	true/false/unknown
Probability Theory	facts	degree of belief 0…1
Fuzzy	degree of truth	degree of belief 0…1

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 $P_{1,2}$

Pros and cons of propositional logic (cont.)

- 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

Example

- Consider the following example:
  - Nam is a student at NLU
  - Every student at NLU is smart
  - Since Nam is a student at NLU, Nam is smart.

- Propositional logic:
  - p: Nam is a student at NLU
  - q: Every student at NLU is smart
  - r: Nam is smart

➔ r cannot be inferred from p and q

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, ...

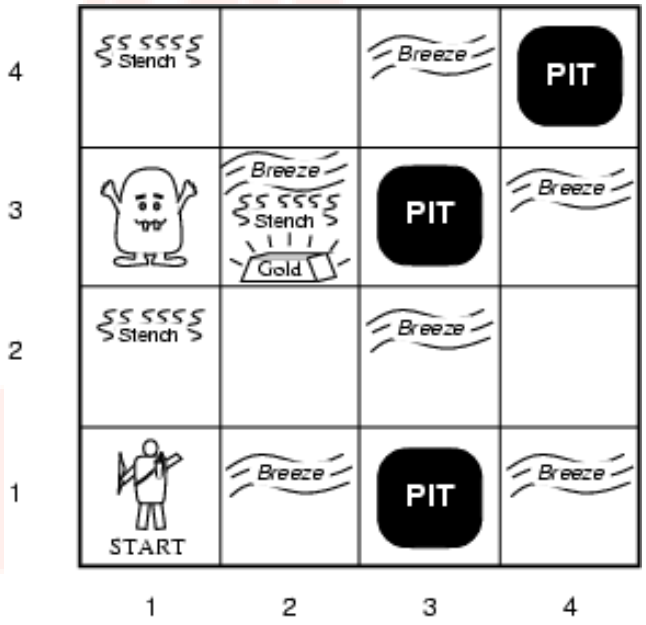
How to identify Objects, Relations, Functions

- Almost any assertion can be thought of as referring to objects and properties or relations
- "One plus two equals three."
  - Objects: one, two, three, one plus two;
  - Relation: equals;
  - Function: plus.
    - "One plus two" is a name for the object that is obtained by applying the function "plus" to the objects "one" and "two."
    - "Three" is another name for this object.

How to identify Objects, Relations, Functions

- Almost any assertion can be thought of as referring to objects and properties or relations

- "Squares neighboring the wumpus are smelly."
  - Objects: wumpus, squares;
  - Property: smelly;
  - Relation: neighboring.



How to identify Objects, Relations, Functions

- Almost any assertion can be thought of as referring to objects and properties or relations
- "Evil King John ruled England in 1200."
  - Objects: John, England, 1200;
  - Relation: ruled;
  - Properties: evil, king.

Relations (Predicates)

- A relation is the set of tuples of objects
  - Brotherhood relationship {<Richard, John>, <John, Richard>}
  - Unary relation, binary relation, …
- Example1: for a given predicate Even(x)
  - Even(2)
  - Event(3)
- Example2: for a given predicate Brother(x, y)
  - Brother(Richard, John)
  - Brother(John, Richard)

Functions

- In English, we use "King John's left leg" rather than giving a name to his leg, where we use "function symbol"

- Example1: LeftLegOf(x), Length(x)
  - LeftLegOf(Richard),
  - LeftLegOf(KingJohn),
  - Length(LeftLegOf(Richard)),
  - Length(LeftLegOf(KingJohn))

First order logic– Syntax in BNF

$$
\begin{array}{rcl}
\textit{Sentence} & \rightarrow & \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} & \rightarrow & \textit{Predicate} \mid \textit{Predicate}(\textit{Term}, \ldots) \mid \textit{Term} = \textit{Term} \\
\textit{ComplexSentence} & \rightarrow & (\ \textit{Sentence}\ ) \mid [\ \textit{Sentence}\ ] \\
 & \mid & \neg\ \textit{Sentence} \\
 & \mid & \textit{Sentence} \wedge \textit{Sentence} \\
 & \mid & \textit{Sentence} \vee \textit{Sentence} \\
 & \mid & \textit{Sentence} \Rightarrow \textit{Sentence} \\
 & \mid & \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
 & \mid & \textit{Quantifier}\ \textit{Variable}, \ldots\ \textit{Sentence} \\
 & & \\
\textit{Term} & \rightarrow & \textit{Function}(\textit{Term}, \ldots) \\
 & \mid & \textit{Constant} \\
 & \mid & \textit{Variable} \\
 & & \\
\textit{Quantifier} & \rightarrow & \forall \mid \exists \\
\textit{Constant} & \rightarrow & A \mid X_1 \mid \textit{John} \mid \cdots \\
\textit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\textit{Predicate} & \rightarrow & \textit{True} \mid \textit{False} \mid \textit{After} \mid \textit{Loves} \mid \textit{Raining} \mid \cdots \\
\textit{Function} & \rightarrow & \textit{Mother} \mid \textit{LeftLeg} \mid \cdots \\
\text{OPERATOR PRECEDENCE} & : & \neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{array}
$$

Atomic sentences

- Atomic sentence = predicate($term_1$, . . . ,$term_n$)

  or $term_1 = term_2$

Term = function($term_1$, . . . ,$term_n$) 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

$\neg S$, $S_1 \wedge S_2$, $S_1 \vee S_2$, $S_1 \rightarrow S_2$, $S_1 \leftrightarrow S_2$

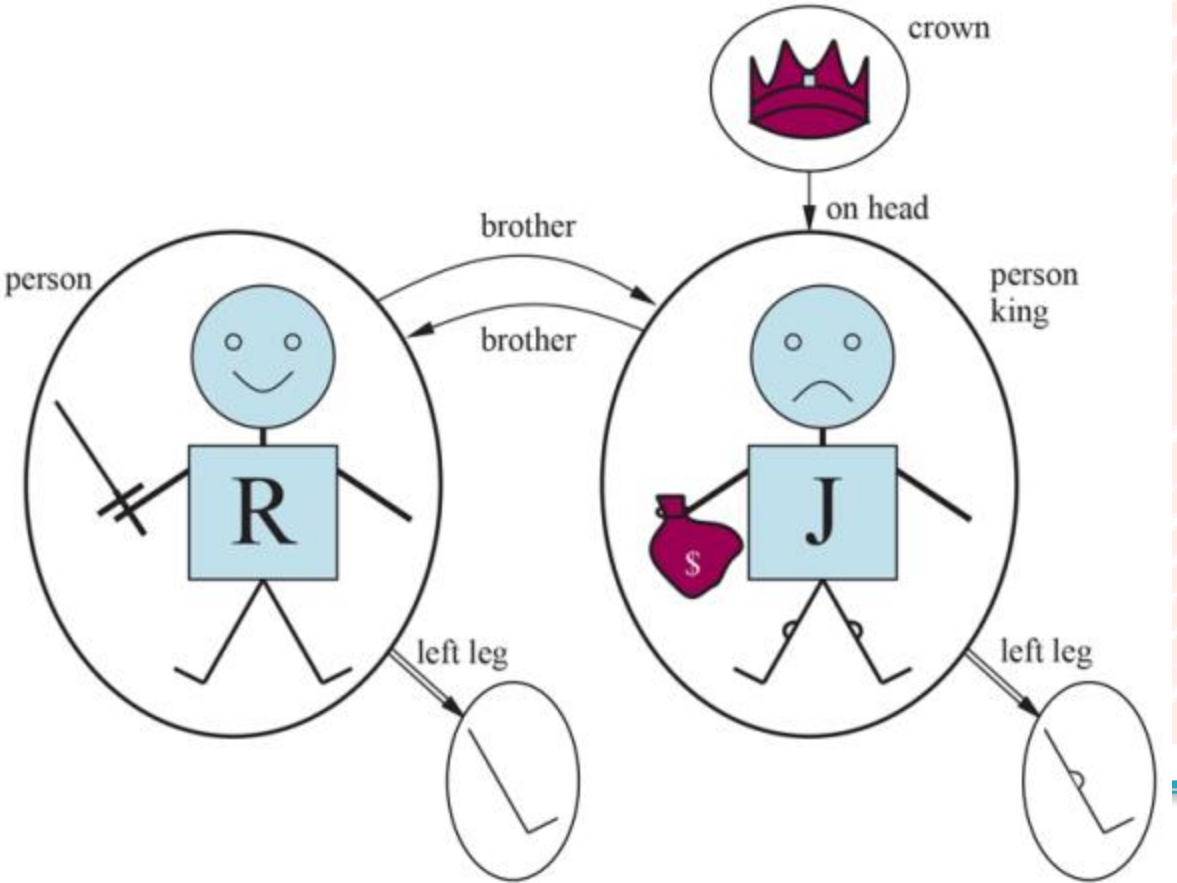
- E.g.
  - Sibling(KingJohn, Richard) $\rightarrow$ Sibling(Richard, KingJohn)
  - $>(1, 2) \vee \leq(1, 2)$
  - $>(1, 2) \wedge \neg>(1, 2)$

Truth in First-order logic

- Sentences are true with respect to a model and an interpretation
- Model contains ≥ 1 objects (domain elements) and relations among them
- Interpretation specifies referents for
  - constant symbols → objects
  - predicate symbols → relations
  - function symbols → functional relations
- An atomic sentence predicate($term_1$; ...; $term_n$) is true
  - Iff the objects referred to by $term_1$; ...; $term_n$ are in the relation referred to by predicate

Models for FOL: Example

- A model containing five objects, two binary relations (brother and on-head), three unary relations (person, king, and crown), and one unary function (left-leg).



Truth example

- Consider the interpretation in which:
  - Richard → Richard the Lionheart
  - John → the evil King John
  - Brother → 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

Universal quantification

- ∀<variables> <sentence>
- Everyone at Berkeley is smart:
- ∀x At(x, Berkeley) → Smart(x)
- ∀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

(At(KingJohn, Berkeley) → Smart(KingJohn))

∧ (At(Richard;Berkeley) → Smart(Richard))

∧ (At(Berkeley;Berkeley) → Smart(Berkeley))

∧ …

A common mistake to avoid

- Typically, $\rightarrow$ is the main connective with $\forall$ (If…then rule)

$$\forall x \; At(x, NLU) \rightarrow Smart(x)$$

- means "Everyone at NLU is smart".
- Common mistake: using $\wedge$ as the main connective with $\forall$:

$$\forall x \; At(x, NLU) \wedge Smart(x)$$

- means "Everyone is at NLU and everyone is smart".

Existential quantification

- ∃<*variables*> <*sentence*>

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

- $\exists x$ *P(x)* 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*
  - (At(Nam, NLU) $\wedge$ Smart(Nam))
  - $\vee$(At(Minh, NLU) $\wedge$ Smart(Minh))
  - $\vee$(At(Lan, NLU) $\wedge$ Smart(Lan))
  - $\vee$ ...

Another common mistake to avoid

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

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

$$\exists x \; At(x, NLU) \rightarrow Smart(x)$$

➔is true if there is anyone who is not at NLU!

Properties of quantfiers

- $\forall x\ \forall y\ P(x,y) \equiv \forall y\ \forall x\ P(x,y)$
- $\exists x\ \exists y\ P(x,y) \equiv \exists y\ \exists x\ P(x,y)$
  - $\forall x\ \forall y$ **Likes(x,y)** Everyone likes everyone.
  - $\forall y\ \forall x$ **Likes(x,y)** Everyone is liked by everyone.
  - Notice: $\exists x\ \exists y$ can be written as $\exists x,y$, (similar for $\forall$)

- $\exists x\ \forall y$ 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“

Properties of quantifiers

1. $(\forall x\ P(x)) \wedge A \equiv \forall x\ (P(x) \wedge A)$
   $(\exists x\ P(x)) \wedge A \equiv \exists x\ (P(x) \wedge A)$
   $(\forall x\ P(x)) \vee A \equiv \forall x\ (P(x) \vee A)$
   $(\exists x\ P(x)) \vee A \equiv \exists x\ (P(x) \vee A)$
   (Where the sentence A not contains x)
2. $\forall x\ P(x) \wedge \forall x\ Q(x) \equiv \forall x\ (P(x) \wedge Q(x))$
   $\exists x\ P(x) \vee \exists x\ Q(x) \equiv \exists x\ (P(x) \vee Q(x))$
3. $\exists x\ P(x) \wedge \exists y\ Q(y) \equiv \exists x\ \exists y\ (P(x) \wedge Q(y))$
   $\forall x\ P(x) \vee \forall y\ Q(y) \equiv \forall x\ \forall y\ (P(x) \vee Q(y))$
4. $\forall x\ P(x) \wedge \exists y\ Q(y) \equiv \forall x\ \exists y\ (P(x) \wedge Q(y))$
   $\forall x\ P(x) \vee \exists y\ Q(y) \equiv \forall x\ \exists y\ (P(x) \vee Q(y))$
5. $\exists x\ (P(x) \rightarrow Q(x)) \equiv \forall x\ P(x) \rightarrow \exists x\ Q(x)$

Negation of quantifiers

- For a given sentence:

$$\forall x \ Likes(x,IceCream)$$

Means "Everyone likes ice cream"

- The negation of the sentence is:

"Not everyone likes ice cream"

- It is equivalent to

"Someone does not like ice cream"

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

- Negation rules:

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

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

Negation Truth Values

- Truth Values of Negated Quantifiers

Statement	True When	False When
$\neg\exists x P(x) \equiv \forall x \neg P(x)$	For every x, $P(x)$ is false.	There is an x for which $P(x)$ is true.
$\neg\forall x P(x) \equiv \exists x \neg P(x)$	There is an x for which $P(x)$ is false.	$P(x)$ is true for every x.

Quantifier duality – Đối ngẫu của lượng từ

- Quantifier duality: each can be expressed using the other
  - S1: $\forall x$ Likes(x,IceCream)

    Everyone likes ice cream
  - S2: $\neg\exists x \neg$Likes(x,IceCream)

    There is no one who does not like ice cream

➔ **S1 and S2: quantifier duality**

  - S3: $\exists x$ Likes(x, IceCream)

    Someone likes ice cream
  - S4: $\neg \forall x \neg$Likes(x, IceCream)

    Not everyone does not like ice cream

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

Equality

- term1 = term2 is true under a given interpretation if and only if term1 and term2 refer to the same object
  - Can be used to state facts about a given function
    - E.g., Father(John) = Henry
  - Can be used with negation to insist that two terms are not the same object
    - E.g., definition of Sibling in terms of Parent:
    - $\forall x,y\ Sibling(x,y) \leftrightarrow [\neg(x = y) \land \exists m,f\ \neg(m = f) \land Parent(m,x) \land Parent(f,x) \land Parent(m,y) \land Parent(f,y)]$

Representing knowledge in FOL

- "Bob is a student."
  - Object can be found from nouns or noun phrases: **Bob**
  - Relation can be found from verbs hoặc verb phrases: **is a student**

Answer: Student(Bob)

- "Deb and Sue are women."
- "Deb and Sue aren't students."
- "Deb and Sue aren’t friends."

Representing knowledge in FOL

Using scope:

- All: anything, everything, whatever, anybody, anyone, everybody, everyone, whoever, …

$$\forall x \ \mathbf{Person}(x) \land \ldots \rightarrow \ldots$$

- Some (at least one): something, somebody, …

$$\exists x \ \mathbf{Person}(x) \land \ldots \land \ldots$$

- No: nothing, nobody, no one, …

$$\neg\exists x \ \mathbf{Person}(x) \land \ldots \land \ldots$$

Representing knowledge in FOL

- *"All stinky shoes are allowed."* (1)
- Answer: $\forall x\ \text{Shoe}(x) \wedge \text{Stinky}(x) \rightarrow \text{Allowed}(x)$

- *"No stinky shoes are allowed."* (2)
- Answer: $\neg\exists x\ \text{Shoe}(x) \wedge \text{Stinky}(x) \wedge \text{Allowed}(x)$

- (2) is equivalent to:
- *"(All) Stinky shoes are not allowed."*
- Answer: $\forall x\ \text{Shoe}(x) \wedge \text{Stinky}(x) \rightarrow \neg\text{Allowed}(x)$

- (2) is equivalent to (1)?

Representing knowledge in FOL

- *"All hoarders collect everything."*
- **Answer:** $\forall x\ (\text{Hoarder}(x) \rightarrow \forall y\ \text{Collects}(x,y))$
  - $\equiv \forall x\ (\neg\text{Hoarder}(x) \vee \forall y\ \text{Collects}(x,y))$
  - $\equiv \forall x\ \forall y\ (\neg\text{Hoarder}(x) \vee \text{Collects}(x,y))$
  - $\equiv \forall x\ \forall y\ (\text{Hoarder}(x) \rightarrow \text{Collects}(x,y))$

Exercise:

- *"(All) Hoarders collect something."*
- **Answer:** $\forall x\ \exists y\ \text{Hoarder}(x) \rightarrow \text{Collects}(x,y)$
- *"Some hoarders collect everything."*
- **Answer:** $\exists x\ \forall y\ \text{Hoarder}(x) \wedge \text{Collects}(x,y)$
- *"Some hoarders collect something."*
- **Answer:** $\exists x\ \exists y\ \text{Hoarder}(x) \wedge \text{Collects}(x,y)$

Representing knowledge in FOL

Complicated sentence:

- *"No hoarders collects anything."*

  Answer: $\neg\exists x\ \forall y\ \text{Hoarder}(x) \wedge \text{Collects}(x,y)$

- Double negation:

  "All hoarders does not collect something."

  Answer: $\forall x\ \exists y\ \text{Hoarder}(x) \rightarrow \neg\text{Collects}(x,y))$

Example:

- *"All hoarders collects nothing."*
- **Answer:** $\forall x\ (\text{Hoarder}(x) \rightarrow \neg\exists y\ \text{Collects}(x, y))$

  $\equiv \forall x\ \forall y\ \text{Hoarder}(x) \rightarrow \neg\text{Collects}(x,y)$

Representing knowledge in FOL

- *"Any good amateur can beat some professional."*
  - $\forall x$ [(x is a good amateur) $\rightarrow$ (x can beat some professional)]
  - (x can beat some professional) **is** $\exists y$ [(y is a professional) $\wedge$ (x can beat y)]

Answer:

$$\forall x\ [(\text{Amateur}(x) \wedge \text{GoodPlayer}(x)) \rightarrow \exists y\ (\text{Professional}(y) \wedge \text{Beat}(x, y))]$$

- *"Some professionals can beat all amateurs."*

Representing knowledge in FOL

Using relation and function:

- *"John's income is 20K"*
  - Function? Income(x)

  Answer: $\text{Income}(\text{John}) = 20K$

- *"There are exactly two shoes"*

  Answer: $\exists x,y\ \text{Shoe}(x) \wedge \text{Shoe}(y) \wedge \neg(x=y) \wedge \forall z\ (\text{Shoe}(z) \rightarrow (x=z) \vee (y=z))$

Representing knowledge in FOL

Using *always*, *sometimes*, *never*

- *"Good people always have friends."*
  means: "*All good people have friends.*"

  $\forall x\ \text{Person}(x) \wedge \text{Good}(x) \rightarrow \exists\ y(\text{Friend}(x,y))$

- *"Busy people sometimes have friends."*
  means : "*Some busy people have friends.*"

  $\exists x\ \text{Person}(x) \wedge \text{Busy}(x) \wedge \exists y(\text{Friend}(x,y))$

- *"Bad people never have friends."*
  means : "*Bad people have no friends.*"

  " $\forall$ x $\text{Person}(x) \wedge \text{Bad}(x) \rightarrow \neg\ \exists\ y(\text{Friend}(x,y))$

  is equivalent to: "*No bad people have friends.*"

  $\neg\ \exists\ x\ \text{Person}(x) \wedge \text{Bad}(x) \wedge \exists\ y(\text{Friend}(x,y))$

Representing knowledge in FOL

- Using *while*, *when*, *whenever*, …

- "Jo always writes home when she is away from home."
- $\forall x\ time(x) \wedge away(Jo,Home,x) \rightarrow writes(Jo,Home,x)$

- "You can fool some of the people all of the time."
- $\exists x\ (person(x) \wedge (\forall t\ time(t) \rightarrow canfool(x,t)))$

- "You can fool all of the people some of the time."
- $\forall x\ person(x) \rightarrow \exists t\ (time(t) \wedge canfool(x,t))$

Representing knowledge in FOL

- "All purple mushrooms are poisonous."
- $\forall x\ \text{mushroom}(x) \wedge \text{purple}(x)) \rightarrow \text{poisonous}(x)$

- "No purple mushroom is poisonous."
- $\neg\exists x\ \text{purple}(x) \wedge \text{mushroom}(x) \wedge \text{poisonous}(x)$
- $\equiv \forall x\ \text{mushroom}(x) \wedge \text{purple}(x)) \rightarrow \neg\text{poisonous}(x)$

- "There are exactly two purple mushrooms."
- $\exists x\ \exists y\ \text{mushroom}(x) \wedge \text{purple}(x) \wedge \text{mushroom}(y) \wedge \text{purple}(y) \wedge \neg(x=y) \wedge (\forall z\ (\text{mushroom}(z) \wedge \text{purple}(z)) \rightarrow ((x=z)\ v\ (y=z)))$

Representing knowledge in FOL

- Example: Kinship domain
- Objects: people
  - John , Mary , Jane , K
- Properties: gender
  - Male (x), Female (x)
- Relations: parenthood, brotherhood, marriage
  - Parent (x, y), Brother (x, y), Spouse (x, y)
- Functions: mother-of (one for each person x)
  - MotherOf (x)

The kinship domain

- Brothers are siblings

$$\forall x,y\ \text{Brother}(x,y) \leftrightarrow \text{Sibling}(x,y)$$

- One's mother is one's female parent

$$\forall m,c\ \text{MotherOf}(c) = m \leftrightarrow (\text{Female}(m) \wedge \text{Parent}(m,c))$$

- "Sibling" is symmetric

$$\forall x, y\ \text{Sibling}(x,y) \leftrightarrow \text{Sibling}(y,x)$$

- Definition of Sibling in terms of Parent:

$$\forall x,y\ \text{Sibling}(x,y) \leftrightarrow [\neg(x = y) \wedge \exists m,f\ \neg(m = f) \wedge \text{Parent}(m,x) \wedge \text{Parent}(f,x) \wedge \text{Parent}(m,y) \wedge \text{Parent}(f,y)]$$

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, ∃a BestAction(a,5))`
  i.e., does the KB entail some best action at t = 5?
  - Answer: Yes, {a / Shoot} ← substitution (binding list)

Interacting with FOL KBs (cont.)

- Given a sentence S and a substitution σ, Sσ denotes the result of plugging σ into S; e.g.,

```
S = Smarter(x,y)
  σ = { x/Hillary, y/Bill }
  Sσ = Smarter(Hillary,Bill)
```

- Ask(KB,S) returns some / all σ such that KB ⊨ Sσ

Knowledge base for the wumpus world

- 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)

- Reflex

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

Deducing hidden properties

- Adjacent properties

$$\forall x,y,a,b \; Adjacent([x,y], [a,b]) \leftrightarrow [a,b] \in \{[x+1,y], [x-1,y], [x,y+1], [x,y-1]\}$$

- Properties of squares:
  - At(Agent ,s,t): the agent is at square s at time t

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

Deducing hidden properties (cont.)

- Squares are breezy near a pit:
  - Diagnostic rule – infer cause from effect
  - $\forall s\ \text{Breezy}(s) \rightarrow \exists r\ \text{Adjacent}(r, s) \wedge \text{Pit}(r)$
  - Causal rule – infer effect from cause
  - $\forall r\ \text{Pit}(r) \rightarrow (\forall s\ \text{Adjacent}(r,s) \rightarrow \text{Breezy}(s))$

Knowledge engineering in FOL

- Identify the task
- Assemble the relevant knowledge
- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain
- Encode a description of the specific problem instance
- Pose queries to the inference procedure and get answers
- Debug the knowledge base

Summary

- First-order logic:
  - objects and relations are semantic primitives
  - syntax: constants, functions, predicates, equality, quantifiers

- Increased expressive power: sufficient to define wumpus world

Exercise

- Biểu diễn các phát biểu sau bằng logic vị từ
  - Hoàng là hàng xóm của chị của Tuấn
  - Không ai vừa là đàn ông vừa là đàn bà
  - Có một câu hỏi trong đề thi này rất khó
  - Tất cả các trẻ em đều thích ăn kem ngoại trừ lúc ốm

Exercise

- Biểu diễn các phát biểu sau bằng logic vị từ
  - Không có 2 người nào có cùng số chứng minh nhân dân
  - Không phải mọi người đều có chị
  - Chỉ có các cây mà cao mới có rễ dài
  - Có một sinh viên ở lớp DH18DT sinh ra ở Hà Nội
  - Tất cả các loài chim đều có thể bay được ngoại trừ chim cánh cụt và đà điểu.

Extra exercise

- For given function MapColor and predicates In(x, y), Borders (x, y),and Country(x)

- Paris and Marseilles are both in France:

$$In(Paris,France) \wedge In(Marseilles,France)$$

Extra exercise

- For given function MapColor and predicates In(x, y), Borders (x, y),and Country(x)

- There is a country that borders both Iraq and Pakistan

- $\exists$ c Country(c) $\wedge$ Border (c, Iraq) $\wedge$ Border (c, Pakistan)

Extra exercise

- For given function MapColor and predicates In(x, y), Borders (x, y),and Country(x)

- All countries that border Ecuador are in South America

- $\forall$ c Country(c) $\wedge$ Border (c, Ecuador) $\Rightarrow$ In(c, SouthAmerica)

Extra exercise

- For given function MapColor and predicates In(x, y), Borders (x, y),and Country(x)

- No region in South America borders any region in Europe

- $\forall$ c, d [In(c, SouthAmerica) $\wedge$ In(d,Europe)$\Rightarrow$ $\neg$Borders(c, d)]

Extra exercise

- For given function MapColor and predicates In(x, y), Borders (x, y),and Country(x)

- No two adjacent countries have the same map color

- $\forall$ x, y (Country(x) $\wedge$ Country(y) $\wedge$ Borders (x, y) $\wedge$ $\neg$(x = y)) $\Rightarrow$ $\neg$(MapColor (x)= MapColor (y))

Extra exercise

- Consider a vocabulary with the following symbols:
- Occupation(p, o): Predicate, Person p has occupation o.
- Customer (p_1,p_2): Predicate, Person p_1 is a customer of person p_2.
- Boss(p_1,p_2): Predicate, Person p_1 is a boss of person p_2.
- Doctor, Surgeon, Lawyer, Actor : Constants denoting occupations.
- Emily, Joe: Constants denoting people.

Extra exercise

- Use these symbols to write the following assertions in first-order logic:
  a) Emily is either a surgeon or a lawyer.
  b) Joe is an actor, but he also holds another job.
  c) All surgeons are doctors.
  d) Joe does not have a lawyer (i.e., is not a customer of any lawyer).
  e) Emily has a boss who is a lawyer.
  f) There exists a lawyer all of whose customers are doctors.
  g) Every surgeon has a lawyer

Extra exercise (sol.)

a) Emily is either a surgeon or a lawyer.
   - Occupation (Emily, Surgeon) $\vee$ Occupation (Emily, Lawyer)

b) Joe is an actor, but he also holds another job.
   - Occupation (Joe, Actor) $\wedge$ $\exists$ p p $\neq$ Actor $\wedge$ Occupation (Joe, p)

c) All surgeons are doctors.
   - $\forall$ p Occupation (p, Surgeon) $\Rightarrow$ Occupation (p, Doctor)

d) Joe does not have a lawyer (i.e., is not a customer of any lawyer).
   - $\neg\exists$ p Customer (Joe, p) $\wedge$ Occupation (p, Lawyer)

Extra exercise (sol.)

e) Emily has a boss who is a lawyer.

- $\exists$ p Boss (p, Emily) $\wedge$ Occupation (p, Lawyer)

f) There exists a lawyer all of whose customers are doctors.

- $\exists$ p Occupation (p,L) $\wedge$ $\forall$ q Customer (q, p) $\Rightarrow$ Occupation (q, Doctor)

g) Every surgeon has a lawyer

- $\forall$ p Occupation (p, Surgeon S) $\Rightarrow \exists$ q Occupation (q, Lawyer) $\wedge$ Customer (p, q)

Group exercise 1

- Express the following sentences in FOL using binary predicates Friend and Enemy:
  1) A friend of a friend is a friend.
  2) An enemy's enemy is a friend.
  3) If two people are friends, then they are not enemies.
  4) Any two people are either enemies or friends.
  5) John has a friend
  6) John has at least two friends.
  7) John has exactly two friends, and everyone else is an enemy.

Group exercise 2

- Express the following sentences in FOL

1) Every student who loves Mary is happy
2) Mary loves everyone except John
3) A vegetarian is someone who eats no meat
4) Các bài hát mà McCartney hát trong album Revolver được viết bởi McCartney
5) Không phải mọi ngôi nhà đều đẹp
6) Mọi ngôi nhà do kiến trúc sư John thiết kế đều đẹp hơn mọi ngôi nhà do kiến trúc sư Mike thiết kế

Inference in first-order logic

Outline

- Reducing first-order inference to propositional inference
- Unification
- Generalized Modus Ponens
- Forward chaining
- Backward chaining
- Resolution

Universal instantiation (UI)

- Every instantiation of a universally quantified sentence is entailed by it (Cụ thể hóa ∀):

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

for any variable v and ground term g

- Example: ***∀x King(x) ∧ Greedy(x) → Evil(x)***
  Applying UI by replacing *x* by ***John***, ***Richard***, … yields:

King(John) ∧ Greedy(John) → Evil(John), {x/John}
King(Richard) ∧ Greedy(Richard) → Evil(Richard), {x/Richard}
King(Father(John)) ∧ Greedy(Father(John)) → Evil(Father(John), {x/Father(John)}

Existential instantiation (EI)

- For any sentence α, variable v, and constant symbol K that does not appear else where in the knowledge base (Cụ thể hóa ∃):

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

- Example:
  - ∃x Crown(x) ∧ OnHead(x, John) applying EI yields: Crown(C1) ∧ OnHead(C1, John)

    provided **C1** is a new constant symbol, called a *Skolem constant*

Reduction to propositional inference

- 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), ...

Reduction (cont.)

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

Reduction (con't)

- THEOREM: Herbrand (1930).
  - If a sentence α is entailed by an FOL KB,
    it is entailed by a finite subset of the propositionalized KB
- IDEA: For n = 0 to ∞ do
  - Create a propositional KB by instantiating with depth-n terms
  - see if α is entailed by this KB
- PROBLEM:
  - works if α is entailed, loops if α is not entailed
- THEOREM: Turing (1936), Church (1936)
  - Entailment for FOL is semi-decidable (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every non-entailed sentence.)

Problems with propositionalization

- Propositionalization seems to generate lots of irrelevant sentences.
- E.g., from: $\forall x\ \text{King}(x) \wedge \text{Greedy}(x) \rightarrow \text{Evil}(x)$

  $\text{King}(\text{John})$

  $\forall y\ \text{Greedy}(y)$

  $\text{Brother}(\text{Richard}, \text{John})$
- It seems obvious that $\text{Evil}(\text{John})$, but propositionalization produces lots of facts such as $\text{Greedy}(\text{Richard})$ that are irrelevant
- With p k-ary (number of arguments) predicates and n constants, there are $p \cdot n^k$ instantiations.

Unification

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

- We can get the inference immediately if we can find a substitution θ such that `King(x) ∧ Greedy(x)` match `King(John) ∧ Greedy(y)`
  Select `θ = {x/John, y/John}` works
- `Unify(p, q) = θ` if `pθ = qθ`

Unification (cont.)

- To unify Knows(John,x) and Knows(y,z)
  θ_1 = {y/John, x/z } or
  θ_2 = {y/John, x/Mary, z/Mary}

- 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 }

Generalized Modus Ponens (GMP)

$$\frac{p'_1, p'_2, \ldots, p'_n, (p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow q)}{\therefore q\theta}$$

where $p_i'\theta = p_i\theta$ for all i

E.g:

p_1': King(John); p1: King(x)

P_2': Greedy(y); p2: Greedy(x); q: Evil(x)

King(x) $\wedge$ Greedy(x) $\rightarrow$ Evil(x)

$\rightarrow$ Evil(John) where θ is {x/John, y/John}

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

Example

- For a given statement:
  - Nam is a student at NLU
  - Every student at NLU is smart
  - Since Nam is a student at NLU, Nam is smart.
- KB: {StudentAtNLU(Nam); $\forall$x SudentAtNLU(x) $\rightarrow$ Smart(x)}

  ➔ Prove: Smart(Nam)
  - Apply UI with {x/Nam} ➔ StudentAtNLU(Nam) $\rightarrow$ Smart(Nam)
  - Apply Modus ponens ➔ Smart(Nam)

Soundness of GMP

- Need to show that

$$p_1', \ldots, p_n', (p_1 \wedge \ldots \wedge p_n \rightarrow q) \models q\theta$$

provided that $p_i'\theta = p_i\theta$ for all i

- Lemma: For any sentence p, we have $p \models p\theta$ by UI
  1. $(p_1 \wedge \ldots \wedge p_n \rightarrow q) \models (p_1 \wedge \ldots \wedge p_n \rightarrow q)\theta = (p_1\theta \wedge \ldots \wedge p_n\theta \rightarrow q\theta)$
  2. $p_1', \ldots, p_n' \models p_1' \wedge \ldots \wedge p_n' \models p_1'\theta \wedge \ldots \wedge p_n'\theta$
  3. From 1 and 2, $q\theta$ follows by ordinary Modus Ponens

Example knowledge base

- The law says that it is a crime for an American to sell weapons to hostile (thù địch) nations.
- The country Nono, an enemy of America, has some missiles (tên lửa), and all of its missiles were sold to it by Colonel West, who is American.
- Prove that Colonel West is a criminal

Example knowledge base (cont.)

- Predicates:
  - American(x): x is an American
  - Weapon(x): x is a weapon
  - Hostile(x): x is a hostile nation
  - Criminal(x): x is a criminal
  - Missile(x): x is a missile
  - Owns(x, y): x owns y
  - Sells(x, y, z): x sells y to z
  - Enemy(x, y): x is an enemy of y
  - Constants: America, Nono, West

Example knowledge base (cont.)

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

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

2. Nono … has some missiles,

   i.e., $\exists x\ Owns(Nono,x) \wedge Missile(x)$: (apply EI)

   $Owns(Nono, M_1) \wedge Missile(M_1)$

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

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

Example knowledge base (cont.)

4. Missiles are weapons:

   $Missile(x) \rightarrow Weapon(x)$

5. An enemy of America counts as "hostile“:

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

6. West, who is American …

   $American(West)$

7. The country Nono, an enemy of America …

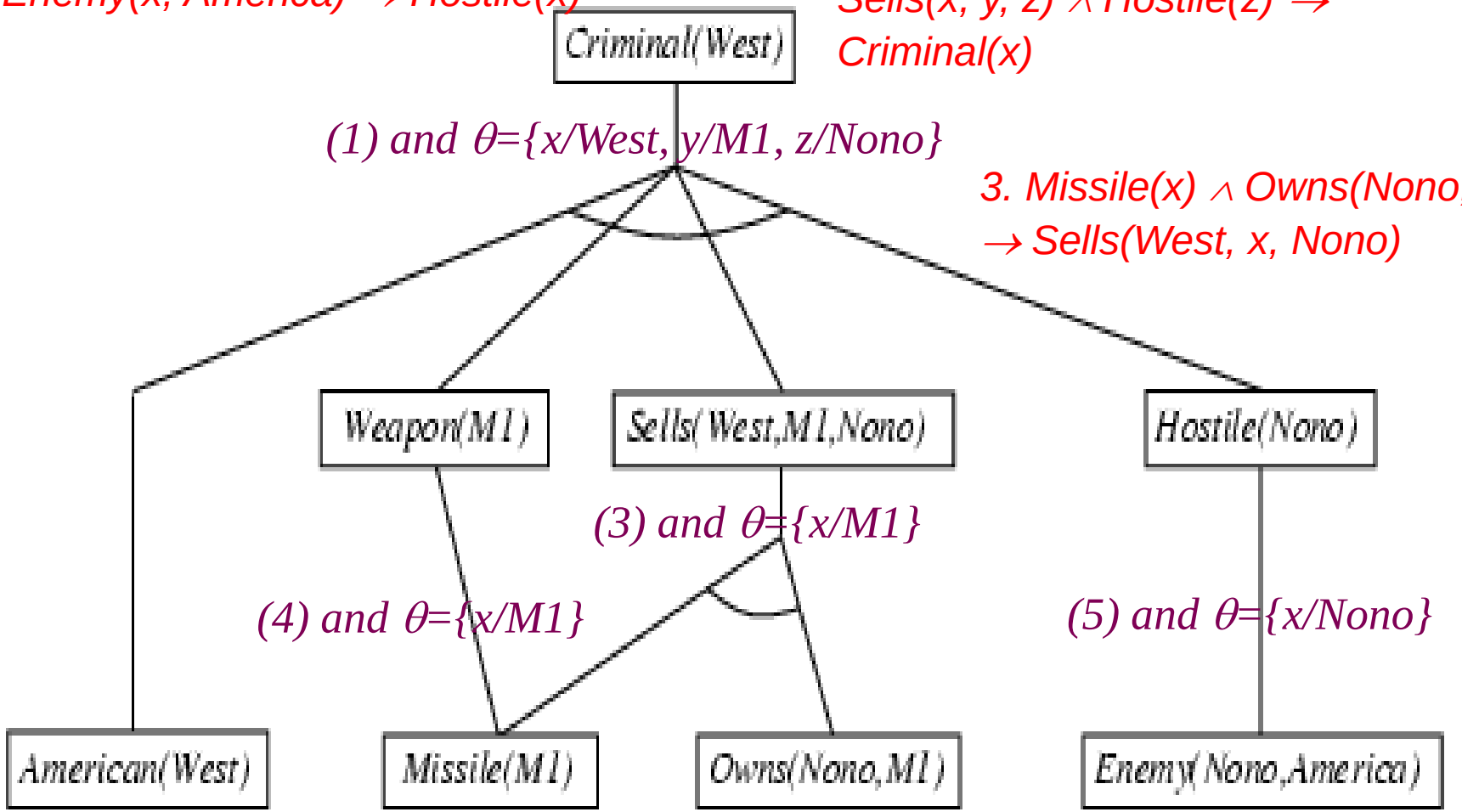
   $Enemy(Nono, America)$

Forward chaining

Forward chaining proof

4. $Missile(x) \rightarrow Weapon(x)$
5. $Enemy(x, America) \rightarrow Hostile(x)$

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



Properties of forward chaining

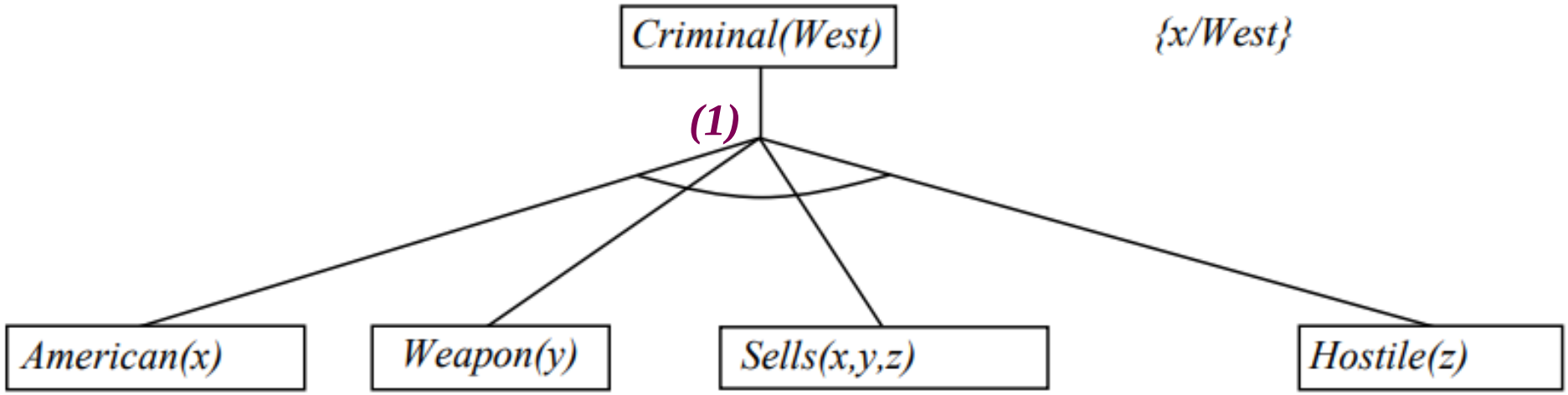
- Sound and complete for first-order definite clauses
- Datalog = first-order definite clauses + no functions
- FC terminates for Datalog in finite number of iterations
- May not terminate in general if α is not entailed
- This is unavoidable: entailment with definite clauses is semidecidable

Backward chaining

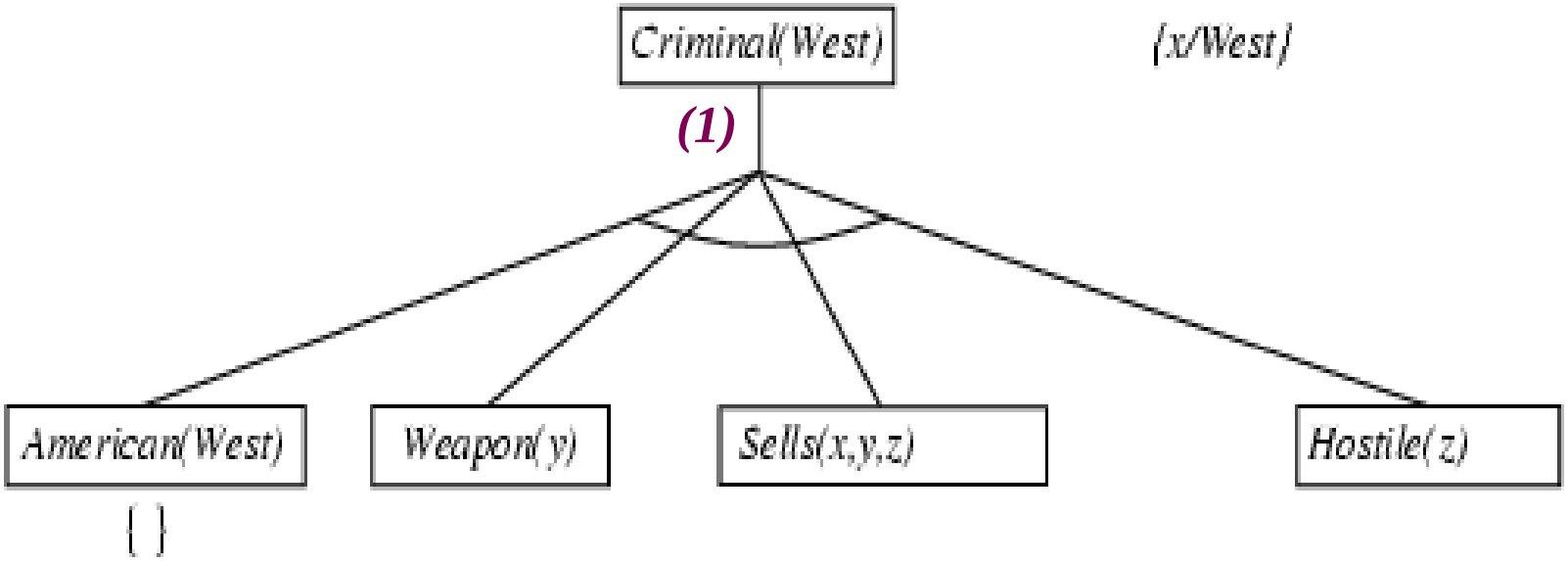
Backward chaining example

Criminal(West)

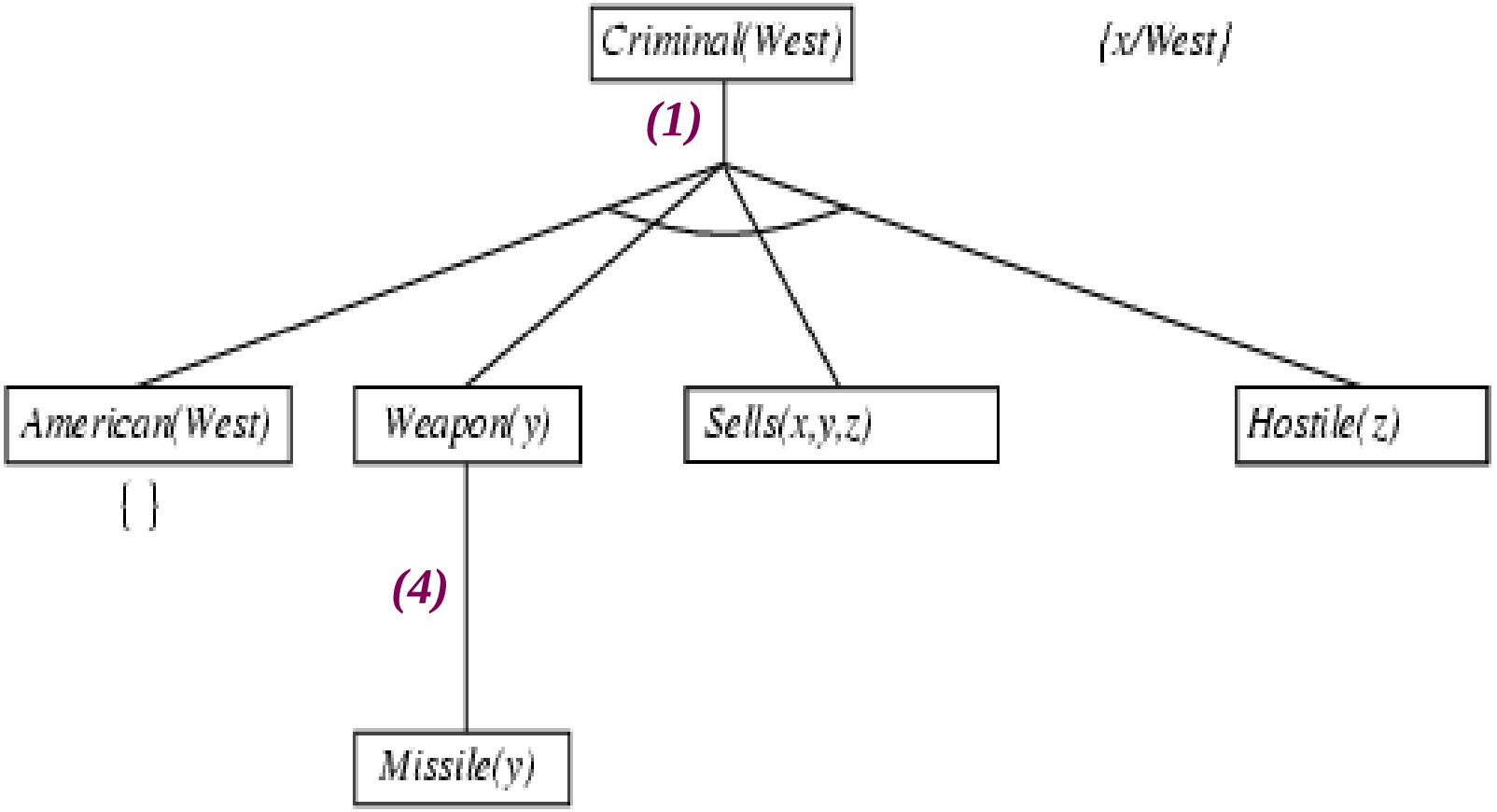
Backward chaining example



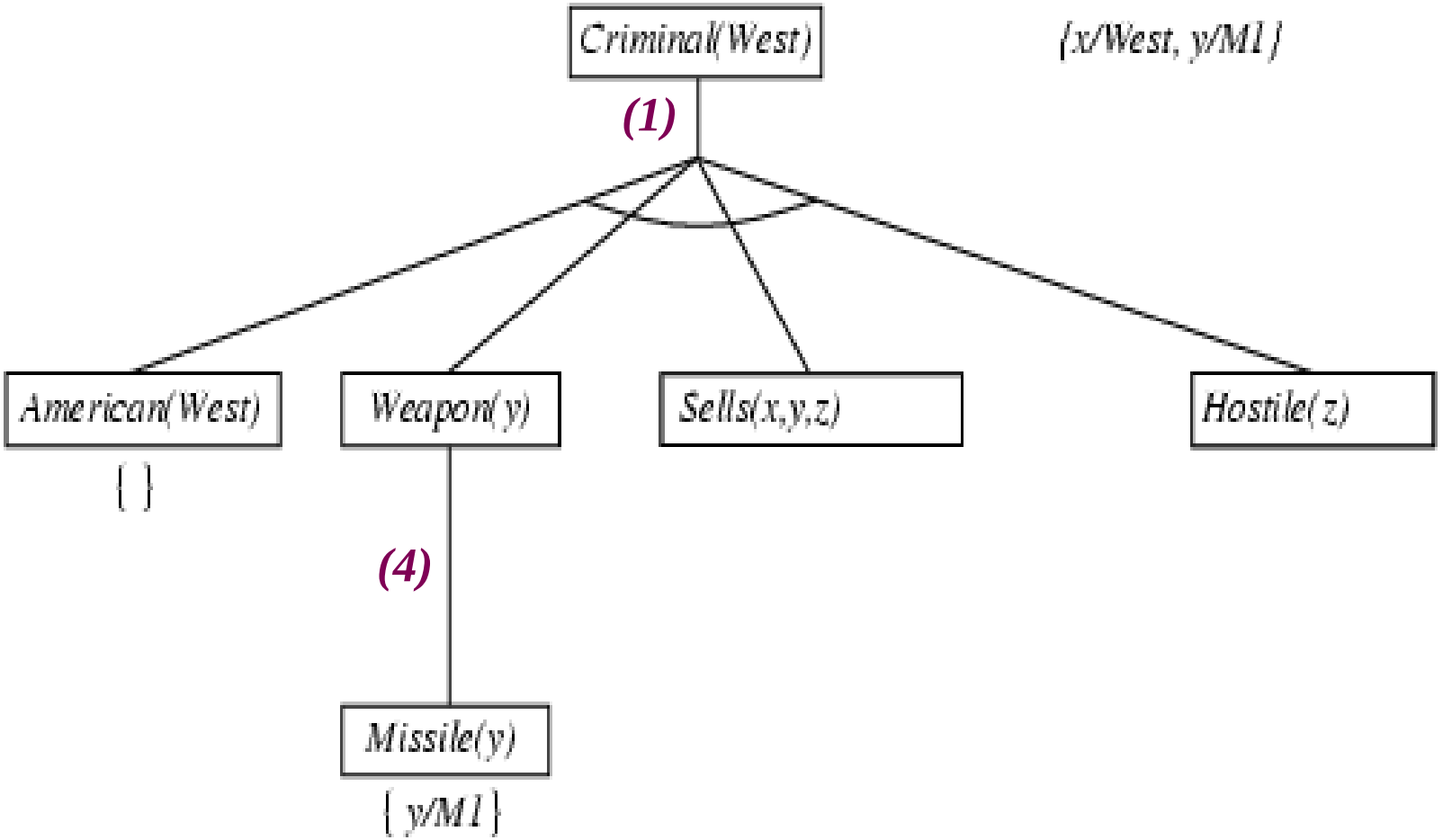
Backward chaining example



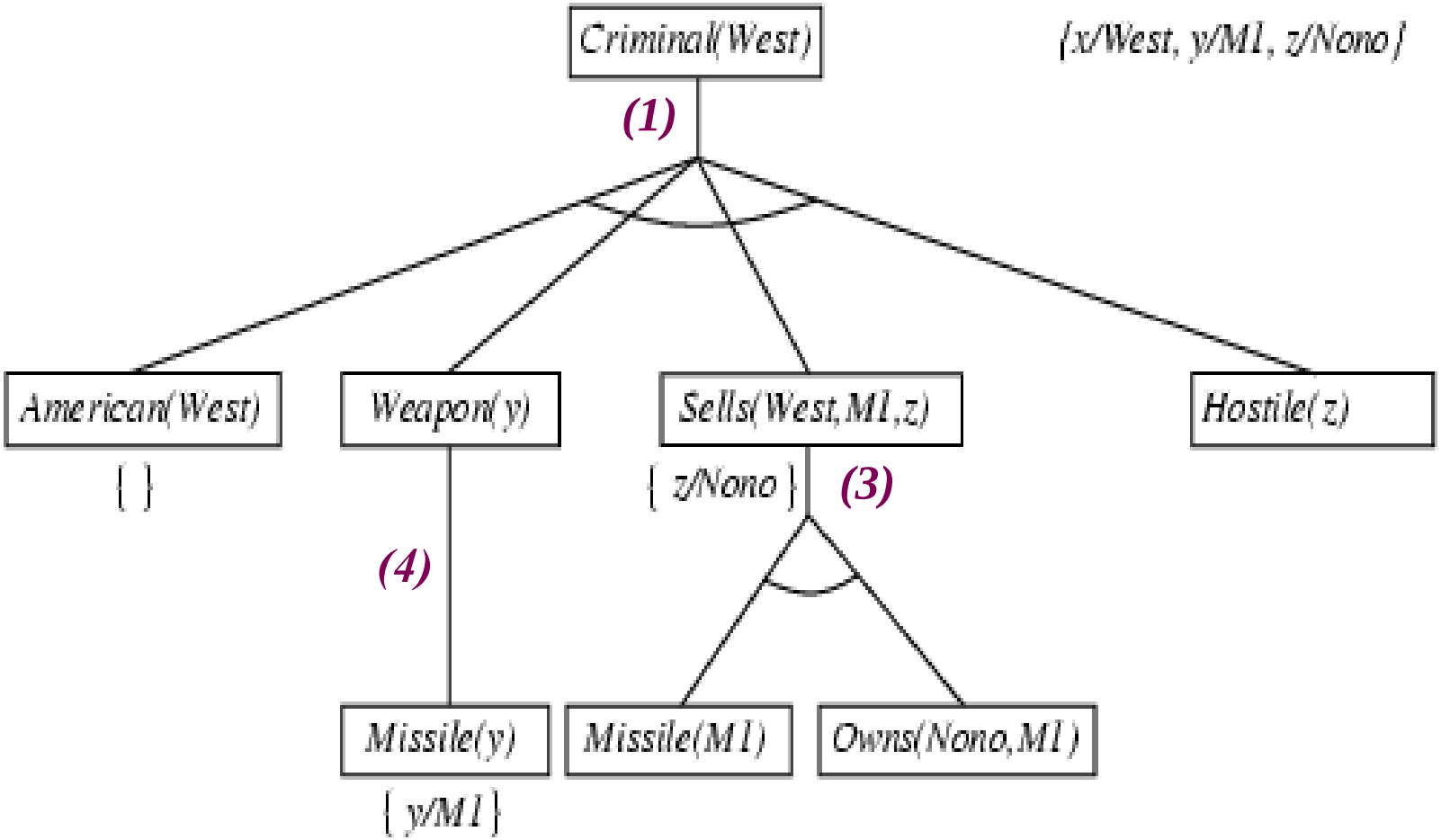
Backward chaining example



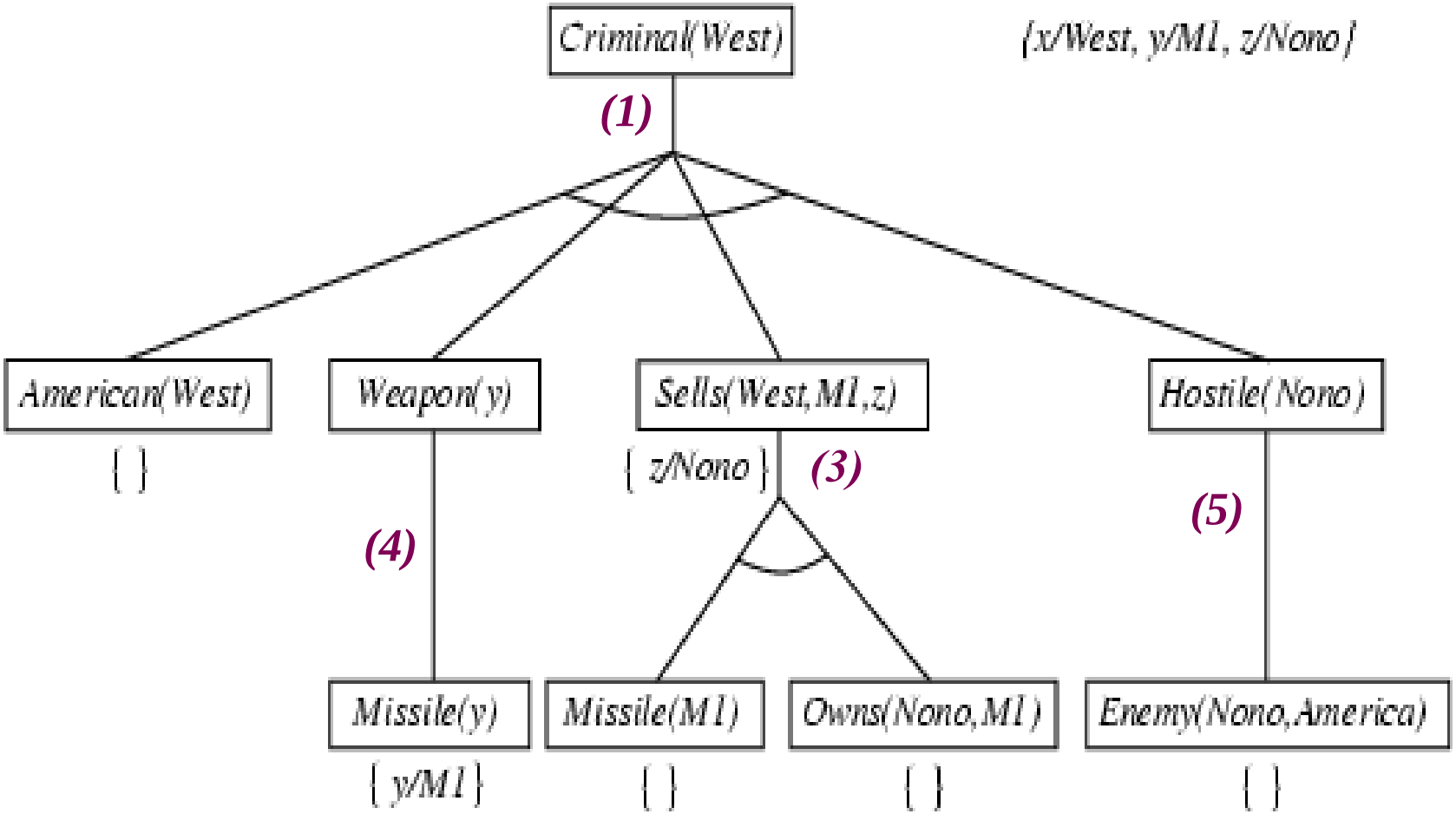
Backward chaining example



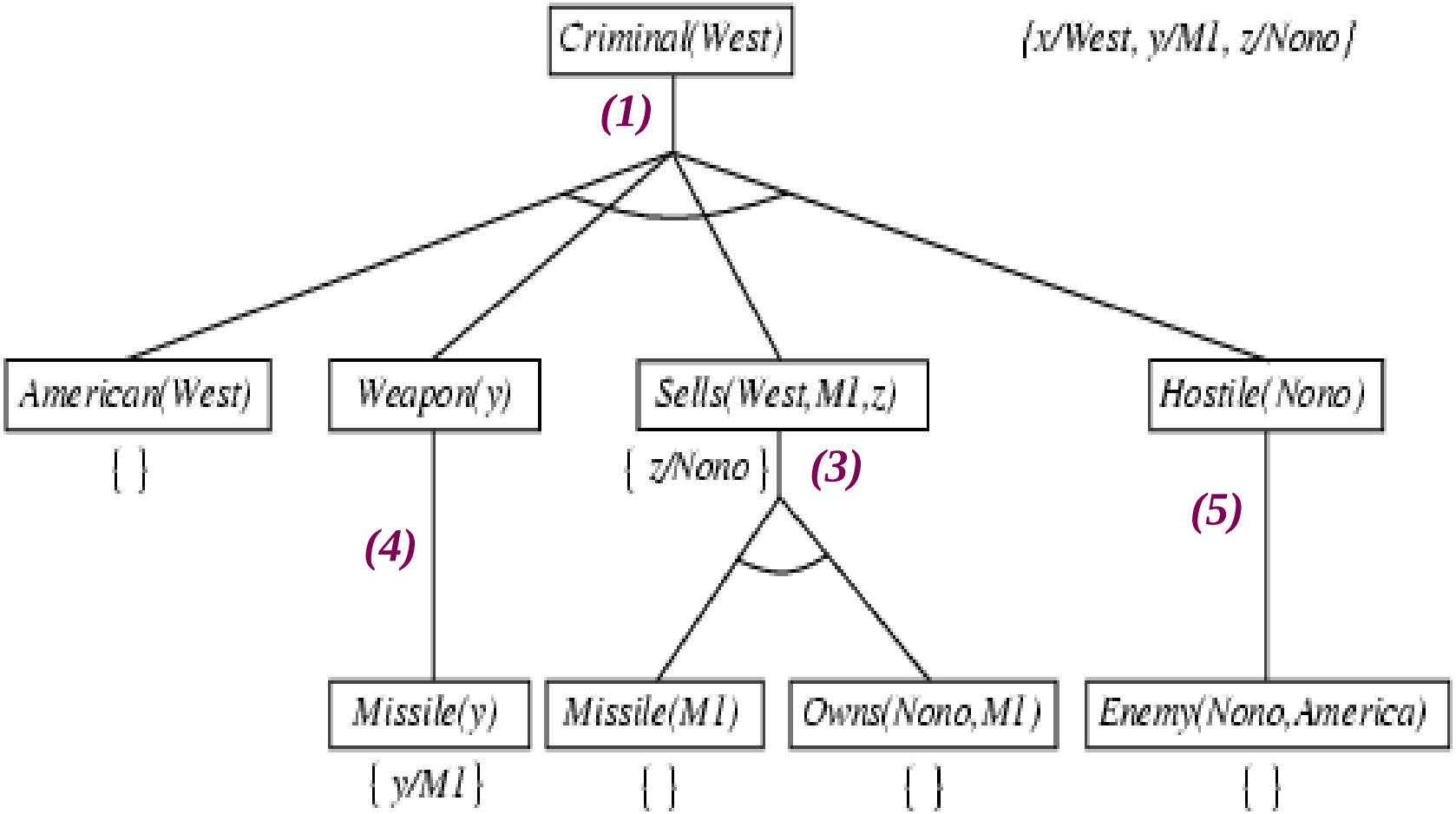
Backward chaining example



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 for logic programming

Resolution

Resolution

- Full first-order version:

$$\frac{\alpha_1 \vee \ldots \vee \alpha_n, \quad \beta_1 \vee \ldots \vee \beta_m}{\therefore \alpha_1 \vee \ldots \vee \alpha_{i-1} \vee \alpha_{i+1} \vee \ldots \vee \alpha_n \vee \beta_1 \vee \ldots \vee \beta_{j-1} \vee \beta_{j+1} \vee \ldots \vee \beta_m}$$

  where **Unify(α_i, $\neg\beta_j$) = θ**

  - The two clauses are assumed to be standardized apart so that they share no variables.

- For example,

$$\frac{\neg Rich(x) \vee Unhappy(x) \qquad Rich(Ken)}{\therefore Unhappy(Ken)}$$

  with θ = {x/Ken}

- Apply resolution steps to CNF(KB $\wedge \neg\alpha$)

Conversion to CNF

Everyone who loves all animals is loved by someone:

$\forall x\ [\forall y\ \text{Animal}(y) \rightarrow \text{Loves}(x, y)] \rightarrow [\exists y\ \text{Loves}(y, x)]$

1. Eliminate biconditionals and implications

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

2. Move $\neg$ inwards (DeMorgan's Laws, double negation):

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

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

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

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

Conversion to CNF (cont.)

3. Standardize variables: each quantifier should use a different one (rename duplicate variables)

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

4. Skolemize: a more general form of existential instantiation.

Each existential variable is replaced by a **Skolem function** of the enclosing universally quantified variables:

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

5. Drop universal quantifiers:

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

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

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

Convert CNF for KB $\wedge \neg\alpha$

1. $\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \rightarrow \text{Criminal}(x)$
   $\equiv \neg(\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z)) \vee \text{Criminal}(x)$
   $\equiv \neg\text{American}(x) \vee \neg\text{Weapon}(y) \vee \neg\text{Sells}(x, y, z) \vee \neg\text{Hostile}(z) \vee \text{Criminal}(x)$

2. $\text{Owns}(\text{Nono}, \text{M1}) \wedge \text{Missile}(\text{M1})$

3. $\text{Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \rightarrow \text{Sells}(\text{West}, x, \text{Nono})$
$\equiv \neg(\text{Missile}(x) \wedge \text{Owns}(\text{Nono}, x)) \vee \text{Sells}(\text{West}, x, \text{Nono})$
$\equiv \neg\text{Missile}(x) \vee \neg\text{Owns}(\text{Nono}, x) \vee \text{Sells}(\text{West}, x, \text{Nono})$

Convert CNF for KB $\wedge \neg\alpha$

4. Missile(x) $\rightarrow$ Weapon(x)
$\equiv \neg$Missile(x) $\vee$ Weapon(x)

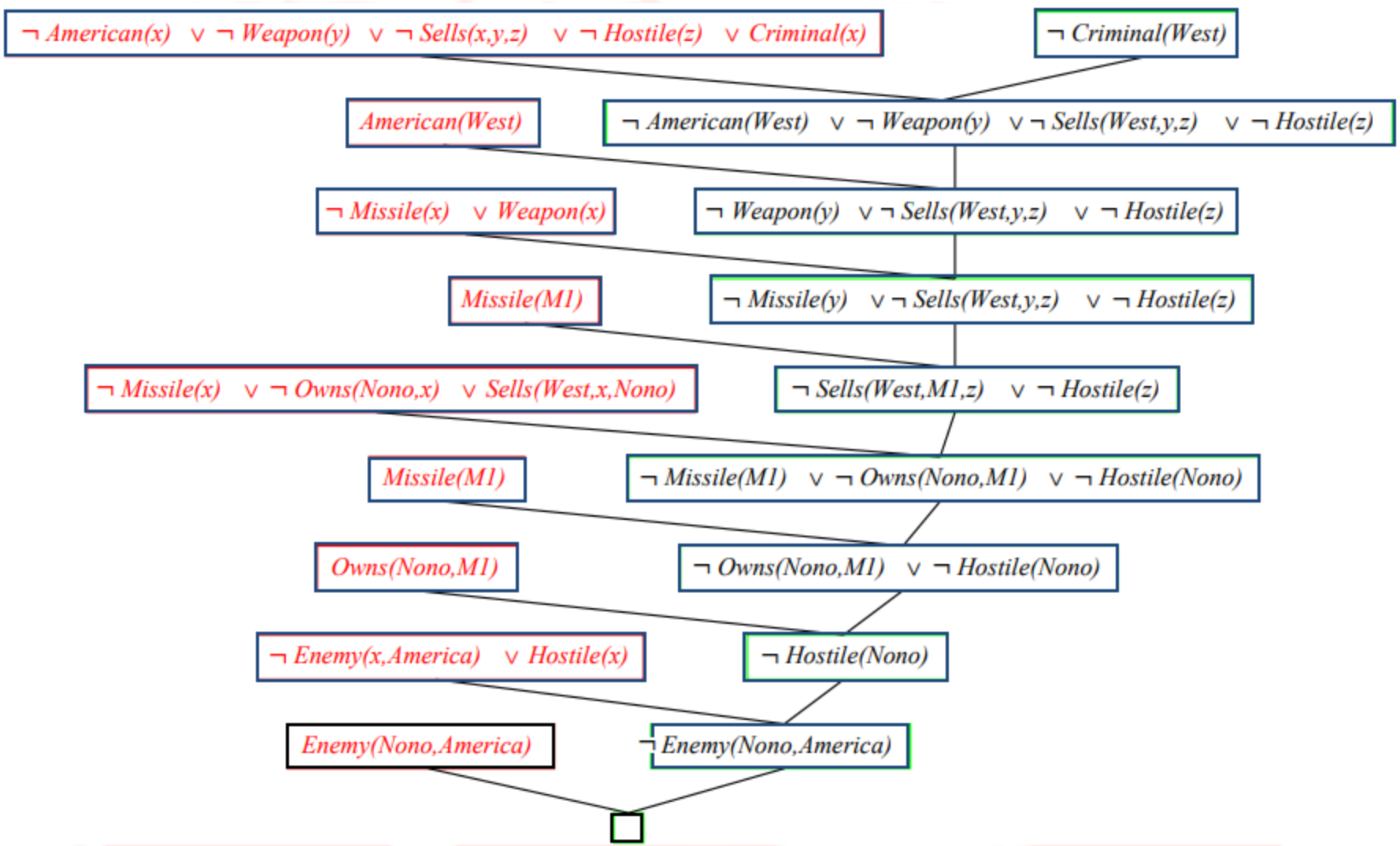
5. Enemy(x, America) $\rightarrow$ Hostile(x)
$\equiv \neg$Enemy(x, America) $\vee$ Hostile(x)

6. American(West)

7. Enemy(Nono, America)

8. $\neg$Criminal(West)

Resolution proof: definite clauses



FACULTY OF INFORMATION TECHNOLOGY

Find a path from S → G