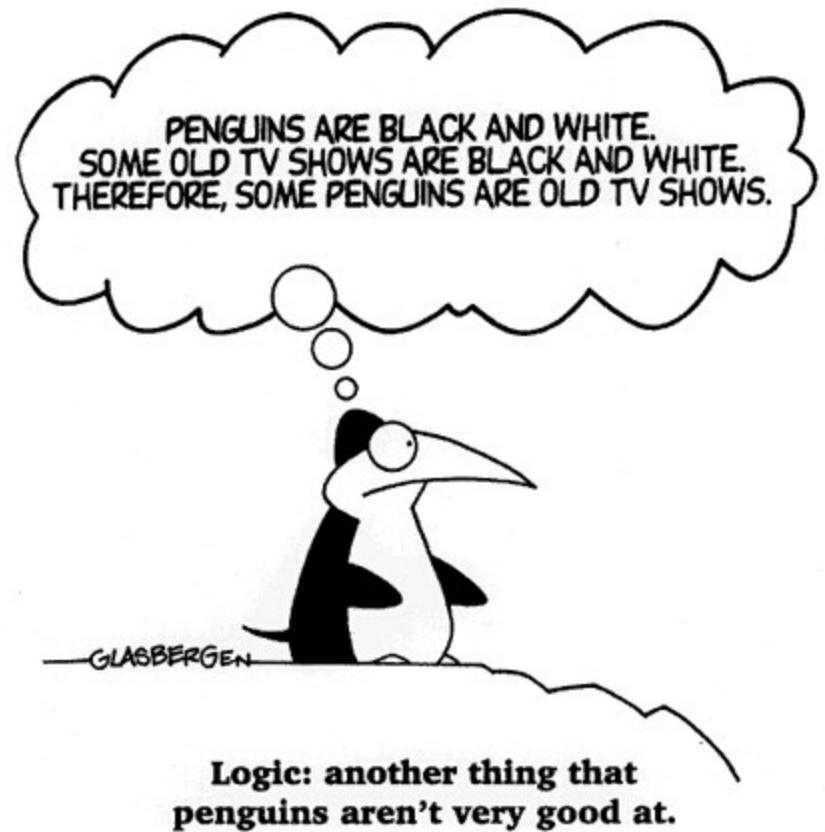


Chapters 8 & 9

First-Order Logic

Dr. Daisy Tang



In which we notice that the world is blessed with objects, some of which are related to other objects, and in which we endeavor to reason about them.

Outline

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

Pros and Cons of Pro. Logic

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

Our Approach

- Adopt **the foundation of propositional logic** – a declarative, compositional semantics that is context-independent and unambiguous – but with more expressive power, **borrowing ideas from natural language** while avoiding its drawbacks
- Important elements in natural language:
 - **Objects** (squares, pits, wumpuses)
 - **Relations** among objects (is adjacent to, is bigger than) or *unary relations* or **properties** (is red, round)
 - **Functions** (father of, best friend of)
 - **First-order logic (FOL)** is built around the above 3 elements

Identify Objects, Relations, Functions

- One plus two equals three.
- Squares neighboring the wumpus are smelly.
- Evil King John ruled England in 1200.

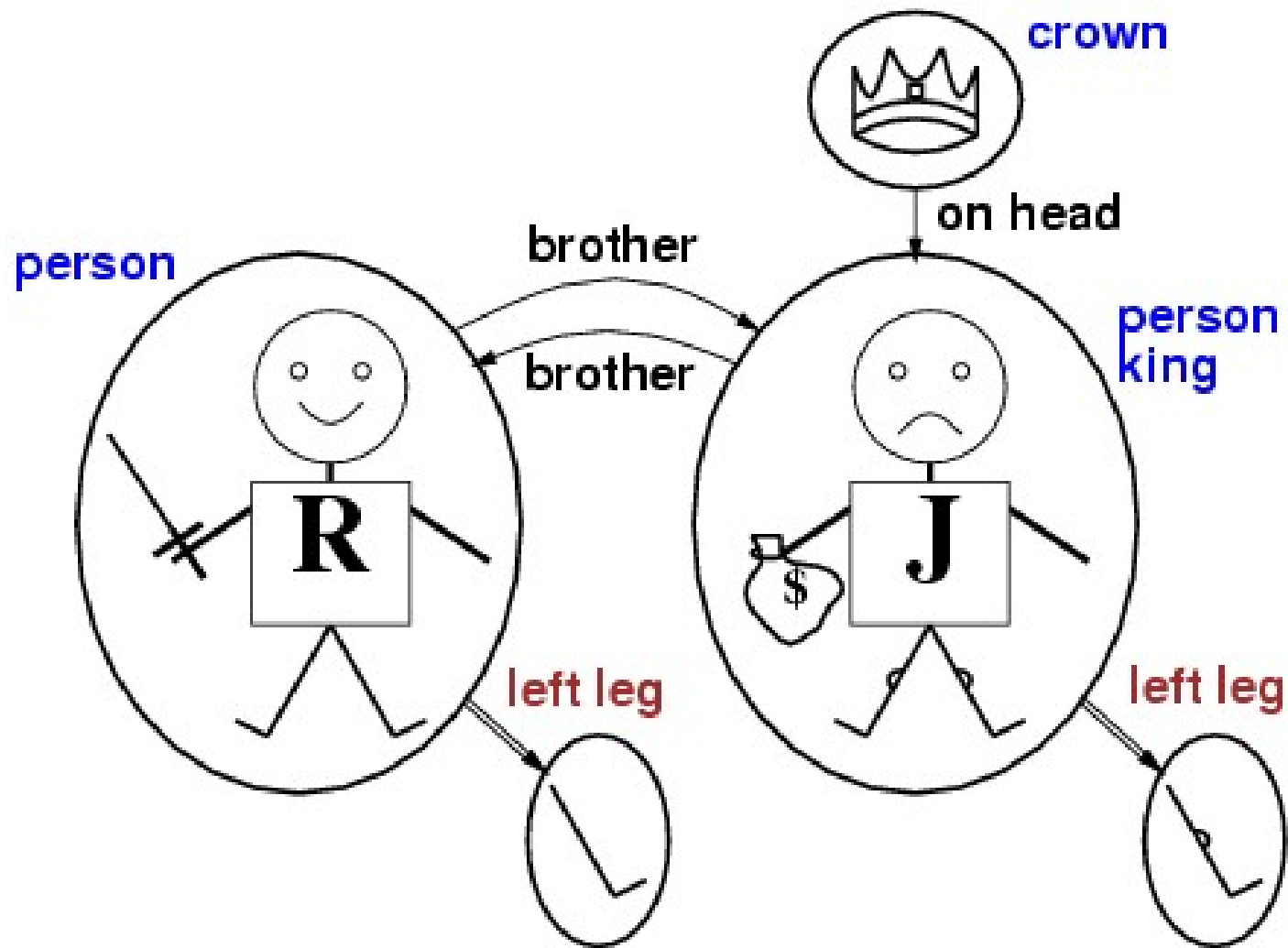
Prop. Logic vs. FOL

- Propositional logic assumes that there are facts that either hold or do not hold
- **FOL assumes more**: the world consists of objects with certain relations among them that do or do not hold
- Other special-purpose logics:
 - **Temporal logic**: facts hold at particular times / T,F,U
 - E.g., “I am always hungry”, “I will eventually be hungry”
 - **Higher-order logic**: views the relations and functions in FOL as objects themselves / T,F,U
 - **Probability theory**: facts / degree of belief [0, 1]
 - **Fuzzy logic**: facts with degree of truth [0, 1] / known interval values
 - E.g., “the temperature is very hot, hot, normal, cold, very cold”

First-Order Logic (FOL)

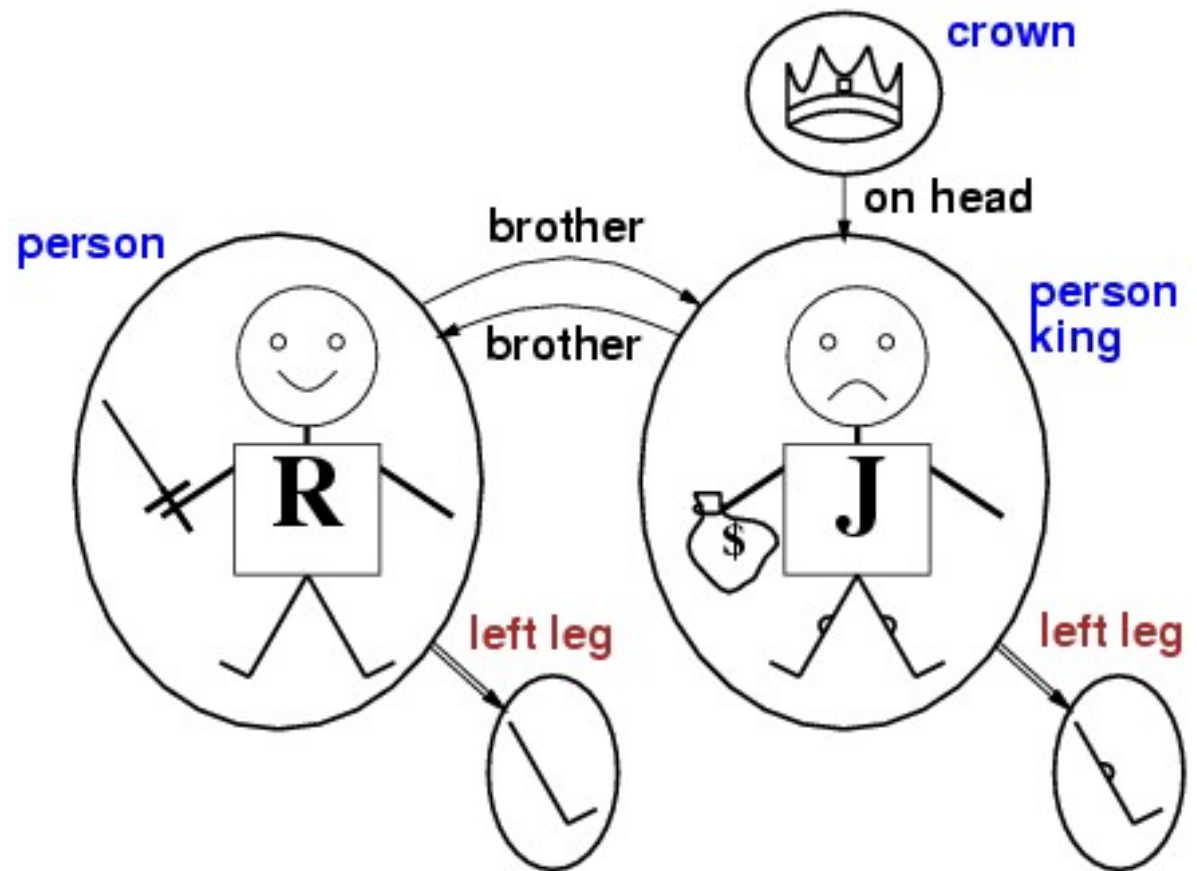
- Whereas propositional logic assumes the world contains **facts**
- First-order logic (like natural language) assumes the world contains
 - **Objects**: people, houses, numbers, colors, baseball games, wars, ...
 - **Relations**: red, round, prime, brother of, bigger than, part of, comes between, ...
 - **Functions**: father of, best friend, one more than, plus, ...

Models for FOL: Example



Example

- Five objects
- Two binary relations
- Three unary relations
- One unary function

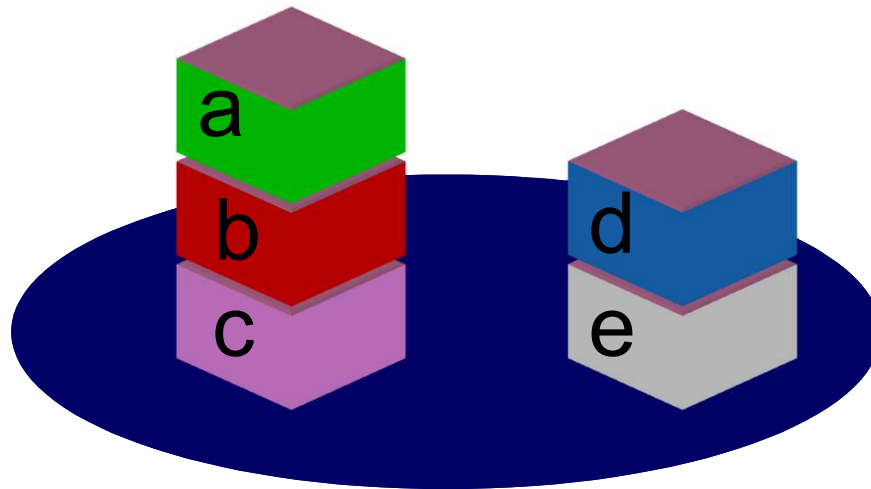


Syntax of FOL: Basic Elements

- Basic symbols: objects (constant symbols), relations (predicate symbols), and functions (functional symbols).
- Constants King John, 2, Wumpus...
- Predicates Brother, $>$, ...
- Functions Sqrt, LeftLegOf, ...
- Variables x, y, a, b, \dots
- Connectives $\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$
- Equality $=$
- Quantifiers \forall, \exists
- A legitimate expression of predicate calculus is called **well-formed formula** (wff), or simply, **sentence**

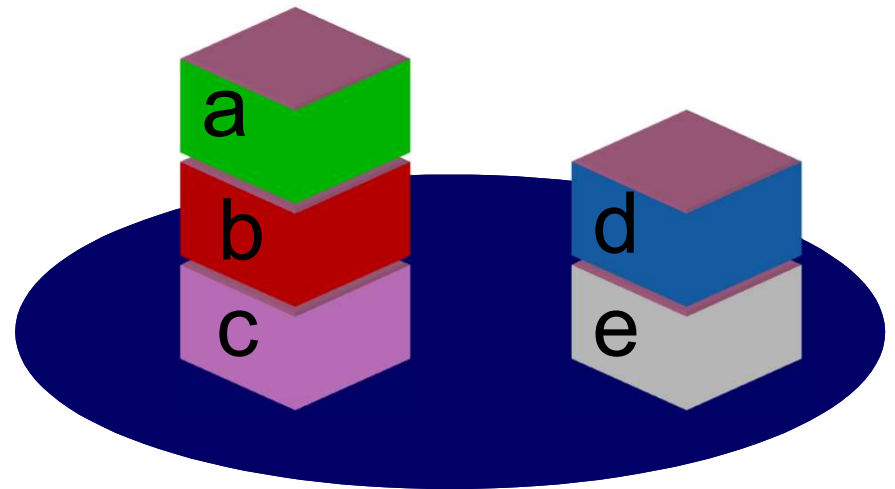
Relations (Predicates)

- A relation is the set of tuples of objects
 - Brotherhood relationship $\{ \langle \text{Richard}, \text{John} \rangle, \langle \text{John}, \text{Richard} \rangle \}$
 - Unary relation, binary relation, ...
- Example: set of blocks $\{a, b, c, d, e\}$
- The "On" relation includes:
 - $\text{On} = \{ \langle a, b \rangle, \langle b, c \rangle, \langle d, e \rangle \}$
 - the predicate $\text{On}(A, B)$ can be interpreted as $\langle a, b \rangle \in \text{On}$.



Functions

- In English, we use “King John’s left leg” rather than giving a name to his leg, where we use “function symbol”
- $\text{hat}(c) = b$
- $\text{hat}(b) = a$
- $\text{hat}(d)$ is not defined



Terms and Atomic Sentences

Atomic sentence = *predicate (term₁, ..., term_n)*
or *term₁ = term₂*

Term = *function (term₁, ..., term_n)*
or *constant or variable*

- Atomic sentence states facts
- Term refers to an object
- For example:
 - *Brother(KingJohn, RichardTheLionheart)*
 - *Length(LeftLegOf(Richard))*
 - *Married(Father(Richard), Mother(John))*

Composite Sentences

- Complex sentences are made from atomic sentences using connectives
- $\neg S, S_1 \wedge S_2, S_1 \vee S_2, S_1 \Rightarrow S_2, S_1 \Leftrightarrow S_2$

For example:

$Sibling(John, Richard) \Rightarrow Sibling(Richard, John)$

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

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

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

Intended Interpretation

- The semantics relate sentences to models to determine truth
- **Interpretation** specifies exactly which objects, relations and functions are referred to by the constant, predicate, and function symbols
 - *Richard* refers to Richard the Lionheart and *John* refers to the evil King John
 - *Brother* refers to the brotherhood relation; *Crown* refer to the set of objects that are crowns
 - *LeftLeg* refers to the “left leg” function
 - There are many other possible interpretations that relate symbols to model; *Richard* refers to the crown
 - If there are 5 objects in the model, how many possible interpretations are there for two symbols *Richard* and *John*?

Intended Interpretation (Con't)

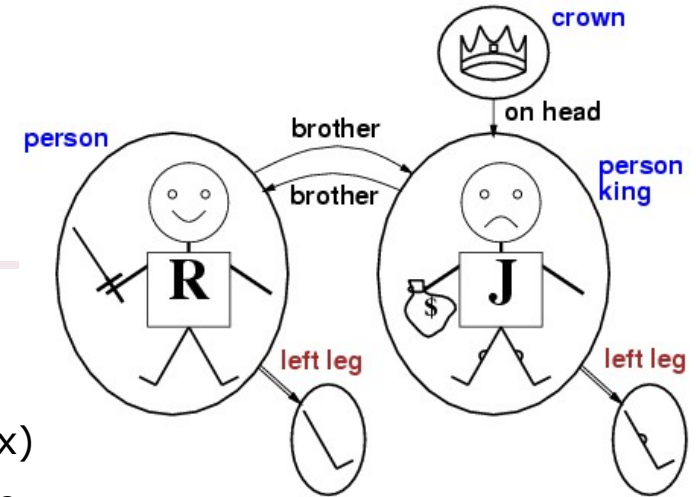
- The **truth of any sentence** is determined by a model and an interpretation for the sentence's model
- The entailment and validity are defined in terms of **all possible models and all possible interpretations**
- The number of domain elements in each model may be unbounded; thus the number of possible models is unbounded
- Checking entailment by enumeration of all possible models is **NOT** doable for FOL

In General

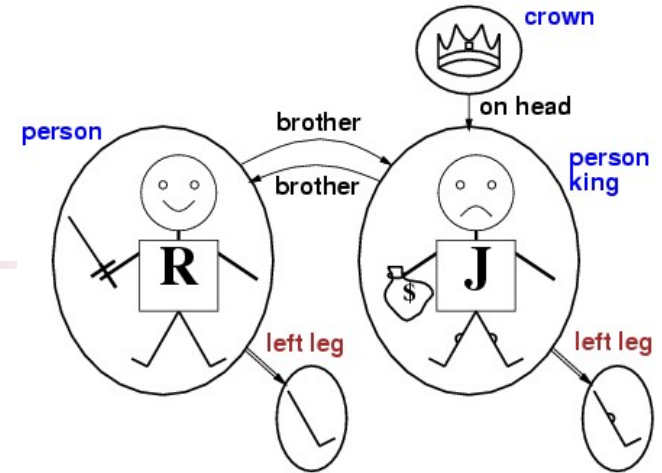
- Sentences are true with respect to a **model** and an **interpretation**
- Model contains objects (**domain elements**) and relations among them
- Interpretation specifies referents for
 - constant symbols** → **objects**
 - predicate symbols** → **relations**
 - function symbols** → **functional relations**
- Once we have a logic that allows objects, it is natural to want to express properties of entire collections of objects, instead of enumerating the objects by name → **Quantifiers**

Universal Quantifier

- Universal quantification (\forall)
 - “All kings are person”: $\forall x \text{ King}(x) \Rightarrow \text{Person}(x)$
 - For all x , if x is a king, then x is a person
- In general, $\forall x P$ is true in a given model under a given interpretation if P is true in all possible extended interpretations
- In the above example, x could be one of the following:
 - Richard, John, Richard’s left leg, John’s left leg, Crown
 - 5 extended interpretations
- A common mistake: $\forall x (\text{King}(x) \wedge \text{Person}(x))$



Existential Quantifier



- Existential quantification (\exists)
 - $\exists x \text{ Crown}(x) \wedge \text{OnHead}(x, \text{John})$
 - There is an x such that x is a crown and x is on the John's head
- In general, $\exists x P$ is true in a given model under a given interpretation if P is true in at least one extended interpretation that assigns x to a domain element
- In the above example, " $\text{Crown}(\text{crown}) \wedge \text{OnHead}(\text{crown}, \text{John})$ " is true
- Common mistake: $\exists x \text{ Crown}(x) \Rightarrow \text{OnHead}(x, \text{John})$

Nested Quantifiers

- Nested quantifiers

- $\forall x \forall y [\text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y)]$
- $\forall x \exists y \text{ Loves}(x, y)$
- $\exists y \forall x \text{ Loves}(x, y)$
- $\exists x \forall y$ is **not** the same as $\forall y \exists x$
- The order of quantification is important

- **Quantifier duality**: each can be expressed using the other

- $\forall x \text{ Likes}(x, \text{IceCream}) \quad \neg \exists x \neg \text{Likes}(x, \text{IceCream})$
- $\exists x \text{ Likes}(x, \text{Broccoli}) \quad \neg \forall x \neg \text{Likes}(x, \text{Broccoli})$

Equality

- $term_1 = term_2$ is true under a given interpretation if and only if $term_1$ and $term_2$ refer to the same object
 - 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*:
 - $\forall x, y \text{ Sibling}(x, y) \Leftrightarrow [\neg(x = y) \wedge \exists m, f \neg (m = f) \wedge \text{Parent}(m, x) \wedge \text{Parent}(f, x) \wedge \text{Parent}(m, y) \wedge \text{Parent}(f, y)]$

Using FOL

- Sentences are added to a KB using **TELL**, which is called **assertions**
- Questions asked using **ASK** are called **queries**
- Any query that is logically entailed by the KB should be answered affirmatively
- The standard form for an answer is a **substitution** or **binding list**, which is a set of **variable/term** pairs

TELL(KB, King(John))

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

ASK(KB, King(John))

ASK(KB, Person(John))

ASK(KB, $\exists x \text{ Person}(x)$) answer : $\{x / \text{John}\}$

Example: The Kinship Domain

- An example KB includes things like:
 - Fact:
 - "Elizabeth is the mother of Charles"
 - "Charles is the father of William"
 - Rules:
 - One's grandmother is the mother of one's parent"
- Object: people
- Unary predicate: Male, Female
- Binary predicate: Son, Spouse, Wife, Husband, Grandparent, Grandchild, Cousin, Aunt, and Uncle
- Function: Mother, Father

Example: The Kinship Domain

One's mom is one's female parent.

$$\forall m, c \text{ Mother}(c) = m \Leftrightarrow \text{Female}(m) \wedge \text{Parent}(m, c)$$

One's husband is one's male spouse.

$$\forall w, h \text{ Husband}(h, w) \Leftrightarrow \text{Male}(h) \wedge \text{Spouse}(h, w)$$

Parent and child are inverse relations.

$$\forall p, c \text{ Parent}(p, c) \Leftrightarrow \text{Child}(c, p)$$

A grandparent is a parent of one's parent.

$$\forall g, c \text{ Grandparent}(g, c) \Leftrightarrow \exists p \text{ Parent}(g, p) \wedge \text{Parent}(p, c)$$

Practice:

Male and female are disjoint categories

A sibling is another child of one's parents

Answer

- Male and female are disjoint categories:
 - $\forall x \text{ Male}(x) \Leftrightarrow \neg \text{Female}(x)$
- A sibling is another child of one's parent:
 - $\forall x, y \text{ Sibling}(x, y) \Leftrightarrow x \neq y \wedge \exists p \text{ Parent}(p, x) \wedge \text{Parent}(p, y)$

The Wumpus World

- A percept is a binary predicate:
 - $\text{Percept}([\text{Stench}, \text{Breeze}, \text{Glitter}, \text{None}, \text{None}], 5)$
- Actions are logical terms:
 - $\text{Turn}(\text{Right}), \text{Turn}(\text{Left}), \text{Forward}, \text{Shoot}, \text{Grab}, \text{Release}, \text{Climb}$
- Query:
 - $\exists a \text{ BestAction}(a, 5)$
 - Answer: $\{a/\text{Grab}\}$
- Perception: percept implies facts:
 - $\forall t, s, b, m, c \text{ Percept}([s, b, \text{Glitter}, m, c], t) \Rightarrow \text{Glitter}(t)$
- Reflex behavior:
 - $\forall t \text{ Glitter}(t) \Rightarrow \text{BestAction}(\text{Grab}, t)$

The Wumpus World

■ The Environment:

- Objects: squares, pits and the wumpus

- $\forall x,y,a,b \text{ Adjacent}([x,y],[a,b]) \Leftrightarrow$

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

- A unary predicate $Pit(x)$
- $\forall s,t \text{ At}(\text{Agent},s,t) \wedge \text{Breeze}(t) \Rightarrow \text{Breezy}(s)$, the agent infers properties of the square from properties of its current percept
- $\text{Breezy}()$ has no time argument
- Having discovered which places are breezy or smelly, not breezy or not smelly, the agent can deduce where the pits are and where the wumpus is

Diagnostic Rules

- **Diagnostic** rule: lead from observed effects to hidden causes
 - If the square is breezy then adjacent square(s) must contain a pit
$$\forall s \text{ Breezy}(s) \Rightarrow \exists r \text{ Adjacent}(r,s) \wedge \text{Pit}(r)$$
 - If the square is not breezy, no adjacent square contains a pit
$$\forall s \neg \text{Breezy}(s) \Rightarrow \neg (\exists r \text{ Adjacent}(r,s) \wedge \text{Pit}(r))$$
 - Combining both:
$$\forall s \text{ Breezy}(s) \Leftrightarrow \exists r \text{ Adjacent}(r,s) \wedge \text{Pit}(r)$$

Causal Rules

- **Causal** rule: some hidden property of the world causes certain percept to be generated
 - A pit causes all adjacent squares to be breezy
 $\forall r \text{ Pit}(r) \Rightarrow [\forall s \text{ Adjacent}(r,s) \Rightarrow \text{Breezy}(s)]$
 - If all squares adjacent to a given square are pitless, the square will not be breezy
 $\forall s [\forall r [\text{Adjacent}(r,s) \Rightarrow \neg \text{Pit}(r)] \Rightarrow \neg \text{Breezy}(s)]$

Summary

- First-order logic:
 - objects and relations are semantic primitives
 - syntax: constants, functions, predicates, equality, quantifiers
- Increased expressive power: sufficient to define wumpus world

Exercises

- Represent the following sentences in FOL
 - Some students took French in spring 2001.
 - Every student who takes French passes it.
 - Only one student took Greek in spring 2001.
 - The best score in Greek is always higher than the best score in French.
- Let the basic vocabulary be as follows:

Student(x)

Takes(x, c, s): student x takes course c in semester s ;

Passes(x, c, s): student x passes course c in semester s ;

Score(x, c, s): the score obtained by student x in course c in semester s ;

$x > y$: x is greater than y ;

F and G : specific French and Greek courses (one could also interpret these sentences as referring to *any* such course, in which case one could use a predicate *Subject*(c, f) meaning that the subject of course c is field f ;

Inference in FOL

- Two ideas:
 - convert the KB to propositional logic and use propositional inference
 - a shortcut that manipulates on first-order sentences directly (resolution, will not be introduced here)

Universal Instantiation

■ Universal instantiation

- infer any sentence by substituting a ground term (a term without variables) for the variable

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

- for any **variable** v and **ground term** g

■ Examples

- $\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$ yields:
- $\text{King}(\text{John}) \wedge \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})$
- $\text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})$
- $\text{King}(\text{Father}(\text{John})) \wedge \text{Greedy}(\text{Father}(\text{John})) \Rightarrow \text{Evil}(\text{Father}(\text{John}))$

Existential Instantiation

- Existential instantiation

- For any sentence α , variable v , and constant symbol k that does **NOT** appear elsewhere in the knowledge base, replace v with k

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

- E.g., $\exists x \text{Crown}(x) \wedge \text{OnHead}(x, \text{John})$ yields:

$$\text{Crown}(C_1) \wedge \text{OnHead}(C_1, \text{John})$$

provided C_1 is a new constant symbol, called a **Skolem constant**

Exercise

- Suppose a knowledge base contains just one sentence, $\exists x \text{ AsHighAs}(x, \text{Everest})$. Which of the following are legitimate results of applying Existential Instantiation?
 - $\text{AsHighAs}(\text{Everest}, \text{Everest})$
 - $\text{AsHighAs}(\text{Kilimanjaro}, \text{Everest})$
 - $\text{AsHighAs}(\text{Kilimajaro}, \text{Everest})$ and $\text{AsHighAs}(\text{BenNevis}, \text{Everest})$

Reduction to Propositional Inference

Suppose the KB contains just the following:

$\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$

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.
- What conclusion can you get?

Reduction Cont'd.

- 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

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

Problems with Propositionalization

- When the KB includes a function symbol, the set of possible ground term substitutions is infinite
 - `Father(Father(...(John)...))`
- Theorem:
 - If a sentence is entailed by the original, first-order KB, then there is a proof involving just a finite subset of the propositionalized KB
 - First instantiation with constant symbols
 - Then terms with depth 1 (`Father(John)`)
 - Then terms with depth 2 ...
 - Until the sentence is entailed

Unification and Lifting

Propositionalization approach is
rather inefficient

A First-Order Inference Rule

- If there is some **substitution** θ that makes the premise of the implication identical to sentences already in the KB, then we assert the conclusion of the implication, after applying θ

$\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$

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

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

What's θ here?

- Sometimes, we need to do is find a substitution θ **both** for the **variables** in the implication and in the sentence to be matched

$\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$

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

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

$\text{Brother}(\text{Richard}, \text{John})$

What's θ here?

Generalized Modus Ponens

- For atomic sentences p_i , p_i' , and q , where there is a substitution θ such that $\text{Subst}(\theta, p_i') = \text{Subst}(\theta, p_i)$, for all i :

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

p_1' is *King(John)*

p_1 is *King(x)*

p_2' is *Greedy(y)*

p_2 is *Greedy(x)*

θ is $\{x/\text{John}, y/\text{John}\}$

q is *Evil(x)*

$\text{Subst}(\theta, q)$ is *Evil(John)*

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

Unification

- GMP is a **lifted version** of Modus Ponens – raises Modus Ponens from propositional to first-order logic
- What's the key **advantage** of GMP over propositionalization?
- Lifted inference rules require finding substitutions that make different logical expressions look identical
- This process is called **unification** and is a key component of all first-order inference algorithms

Unification

- The unify algorithm takes two sentences and returns a unifier for them if one exists:
 - $\text{UNIFY}(p, q) = \theta$ where $\text{Subst}(\theta, p) = \text{Subst}(\theta, q)$

p	q	θ
Knows(John,x)	Knows(John,Jane)	
Knows(John,x)	Knows(y,OJ)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

Unification

- The unify algorithm takes two sentences and returns a unifier for them if one exists:
 - $\text{UNIFY}(p, q) = \theta$ where $\text{Subst}(\theta, p) = \text{Subst}(\theta, q)$

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

Unification

- The unify algorithm takes two sentences and returns a unifier for them if one exists:
 - $\text{UNIFY}(p, q) = \theta$ where $\text{Subst}(\theta, p) = \text{Subst}(\theta, q)$

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

Unification

- The unify algorithm takes two sentences and returns a unifier for them if one exists:
 - $\text{UNIFY}(p, q) = \theta$ where $\text{Subst}(\theta, p) = \text{Subst}(\theta, q)$

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

Unification

- The unify algorithm takes two sentences and returns a unifier for them if one exists:
 - $\text{UNIFY}(p, q) = \theta$ where $\text{Subst}(\theta, p) = \text{Subst}(\theta, q)$

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

Standardizing apart: renaming variables to avoid name clashes

$\text{UNIFY}(\text{Knows}(\text{John}, x), \text{Knows}(z, \text{OJ})) = \{x/\text{OJ}, z/\text{John}\}$

Unification Contd.

- To unify $Knows(John, x)$ and $Knows(y, z)$,
 $\theta = \{ y/John, x/z \}$ or $\theta = \{ y/John, x/John, z/John \}$
- The first unifier is **more general** than the second
- There is a single **most general unifier** (MGU)
- $MGU = \{ y/John, x/z \}$

Exercise

- For each pair of atomic sentences, give the most general unifier if it exists:
 - $P(A, B, B), P(x, y, z)$
 - $Q(y, G(A, B)), Q(G(x, x), y)$
 - $\text{Older}(\text{Father}(y), y), \text{Older}(\text{Father}(x), \text{John})$
 - $\text{Knows}(\text{Father}(y), y), \text{Knows}(x, x)$

Example Knowledge Base

- The law says that it is a crime for an *American* to sell weapons to hostile nations. The country *Nono*, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel *West*, who is American.
- Prove that Colonel West is a criminal
 - $\text{American}(x)$: x is an American
 - $\text{Weapon}(x)$: x is a weapon
 - $\text{Hostile}(x)$: x is a hostile nation
 - $\text{Criminal}(x)$: x is a criminal
 - $\text{Missile}(x)$: x is a missile
 - $\text{Owns}(x, y)$: x owns y
 - $\text{Sells}(x, y, z)$: x sells y to z
 - $\text{Enemy}(x, y)$: x is an enemy of y
 - Constants: America, Nono, West

Example Knowledge Base

First-Order Definite Clauses

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

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

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

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

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

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

Missiles are weapons:

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

An enemy of America counts as "hostile":

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

West, who is American ...

$American(West)$

The country Nono, an enemy of America ...

$Enemy(Nono, America)$

Forward Chaining

- Starting from known facts, it triggers all the rules whose premises are satisfied, adding their conclusions to the known facts. This process will continue until query is answered.
- When a new fact P is added to the KB:
 - For each rule such that P unifies with a premise
 - if the other premises are already known
 - then add the conclusion to the KB and continue chaining
- Forward chaining is data-driven:
 - inferring properties and categories from percepts

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

Forward Chaining Proof

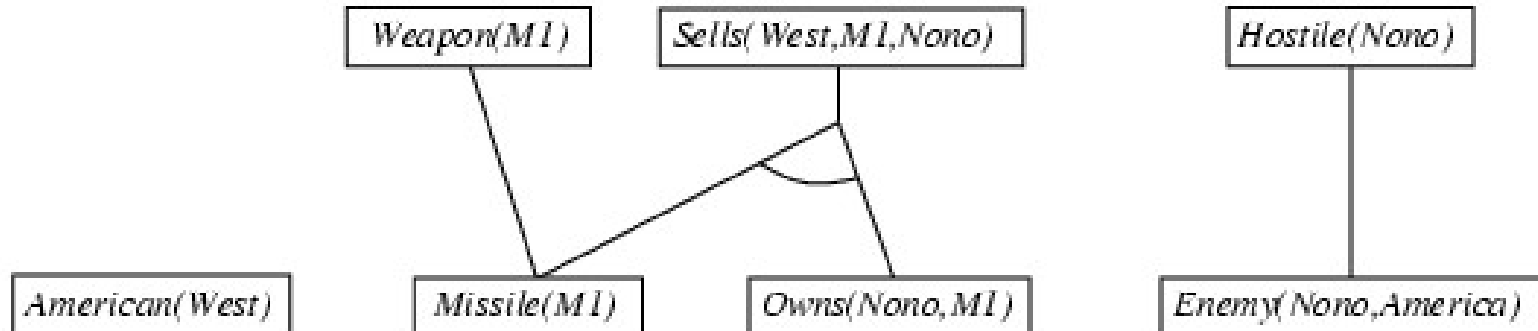
American(West)

Missile(M1)

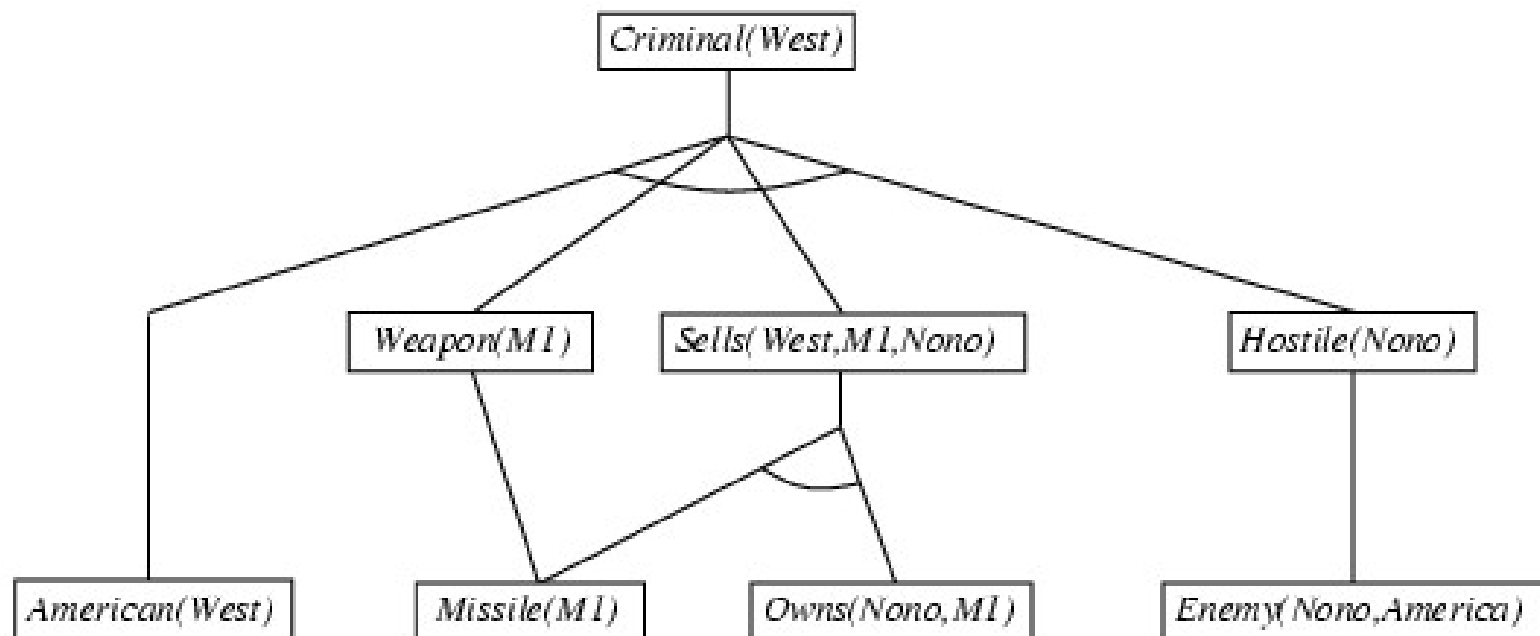
Owns(Nono,M1)

Enemy(Nono,America)

Forward Chaining Proof



Forward Chaining Proof



Analysis of Forward Chaining

- A fixed point of the inference process can be reached
- The algorithm is sound and complete
- Its efficiency can be improved

FC Algorithm Analysis

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

Sources of Complexity

- 1. “Inner loop” involves finding **all possible unifiers** such that the premise of a rule unifies with facts in the KB
 - Often called “**pattern matching**”, very expensive
- 2. The algorithm **rechecks every rule** on every iteration to see whether its premises are met
- 3. It might generate many **facts** that are **irrelevant** to the goal

Matching Rules Against Known Facts

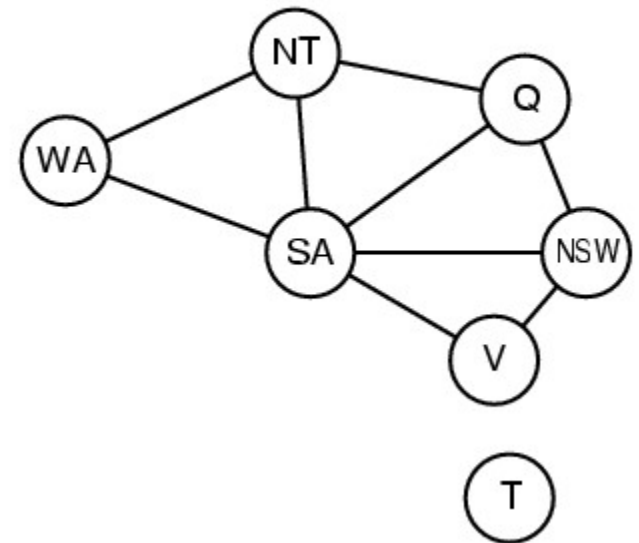
- Consider a rule:
 - $Owns(Nono, x) \wedge Missile(x) \Rightarrow Sells(West, x, Nono)$
 - We find all the objects owned by Nono in constant time per object;
 - Then, for each object, we check whether it's a missile.
 - If more objects and very few missiles → inefficient
 - Conjunct ordering problem: NP-hard
 - Heuristics?

Pattern Matching and CSP

- The **most constrained variable heuristic** used for CSPs would suggest ordering the conjuncts to look for missiles first if there are fewer missiles than objects owned by Nono
- We can actually express every finite-domain CSP as a single definite clause together with some associated ground facts

$\text{Diff}(\text{wa}, \text{nt}) \wedge \text{Diff}(\text{wa}, \text{sa}) \wedge$
 $\text{Diff}(\text{nt}, \text{q}) \wedge \text{Diff}(\text{nt}, \text{sa}) \wedge$
 $\text{Diff}(\text{nsw}, \text{v}) \wedge \text{Diff}(\text{nsw}, \text{v}) \wedge$
 $\text{Diff}(\text{v}, \text{sa}) \rightarrow \text{Colorable}()$

$\text{Diff}(\text{R}, \text{B}) \quad \text{Diff}(\text{R}, \text{G})$
 $\text{Diff}(\text{G}, \text{R}) \quad \text{Diff}(\text{G}, \text{B})$
 $\text{Diff}(\text{B}, \text{R}) \quad \text{Diff}(\text{B}, \text{G})$



Incremental Forward Chaining

- Observation:
 - Every new fact inferred on iteration t must be derived from at least one new fact inferred on iteration $t-1$
- Modification:
 - At iteration t , we check a rule only if its premise includes a conjunct p_i that unifies with a fact p_i' newly inferred at iteration $t-1$
 - Many real systems operate in an “update” mode wherein **forward chaining occurs in response to each new fact** that is TOLD to the system

Irrelevant Facts

- FC makes all allowable inferences based on known facts, even if they are irrelevant to the goal at hand
- Similar to FC in propositional context
- Solution?

Backward Chaining

- $p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q$
- When a query q is examined:
 - if a matching fact q' is known, return the unifier
 - for each rule whose consequent q' matches q
 - attempt to prove each **premise** of the rule by backward chaining
- Backward chaining is the basis for “logic programming,” e.g., **Prolog**

Backward Chaining Algorithm

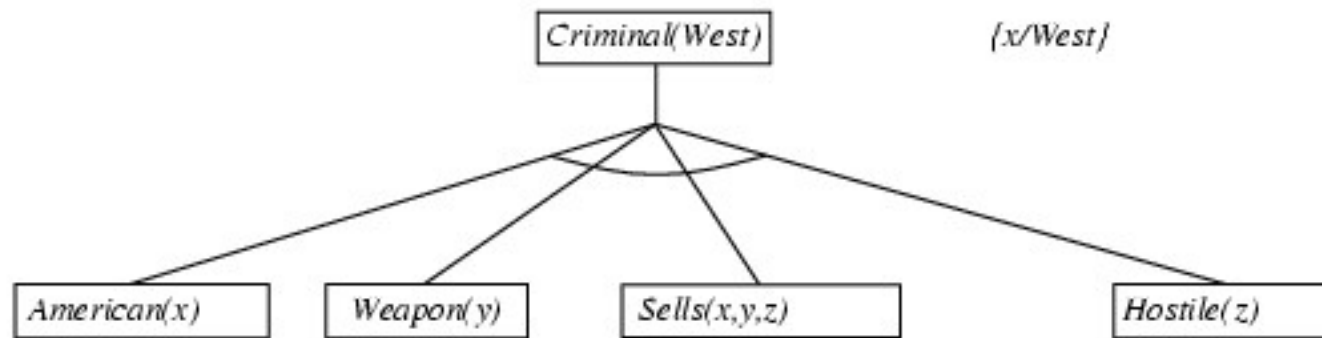
```
function FOL-BC-ASK( $KB, goals, \theta$ ) returns a set of substitutions
  inputs:  $KB$ , a knowledge base
            $goals$ , a list of conjuncts forming a query
            $\theta$ , the current substitution, initially the empty substitution  $\{ \}$ 
  local variables:  $ans$ , a set of substitutions,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

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

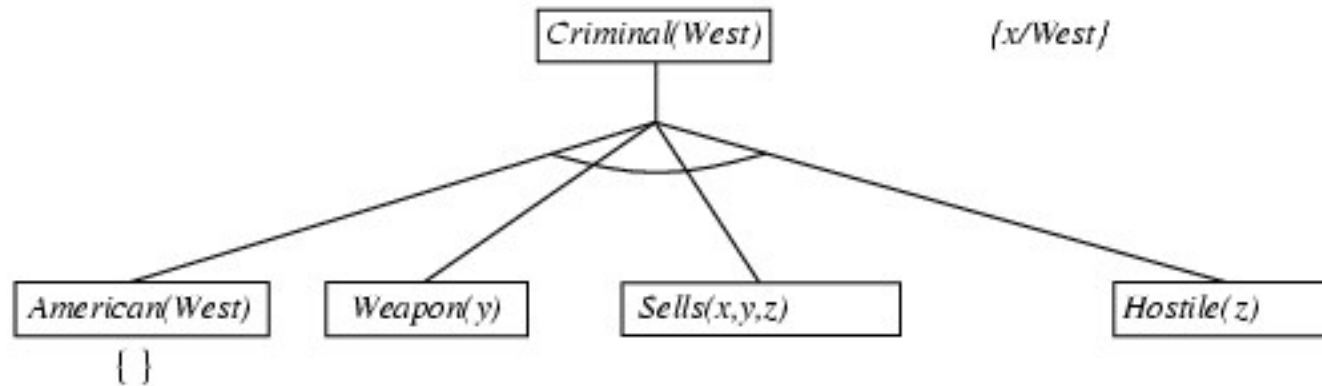
Backward Chaining Example

Criminal(West)

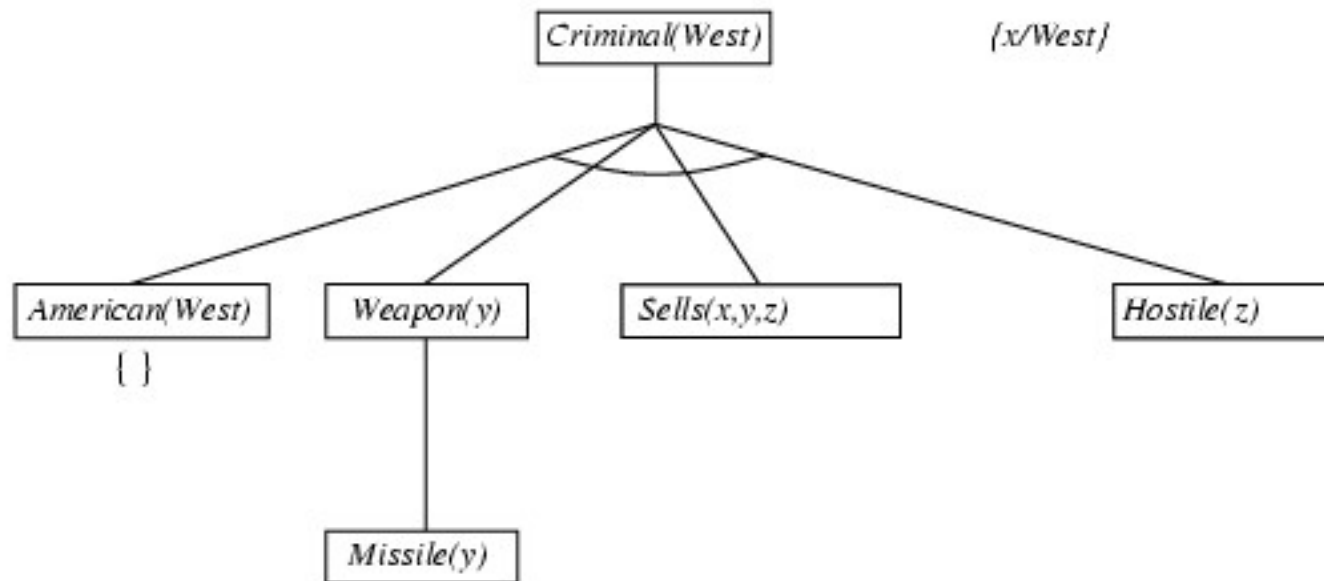
Backward Chaining Example



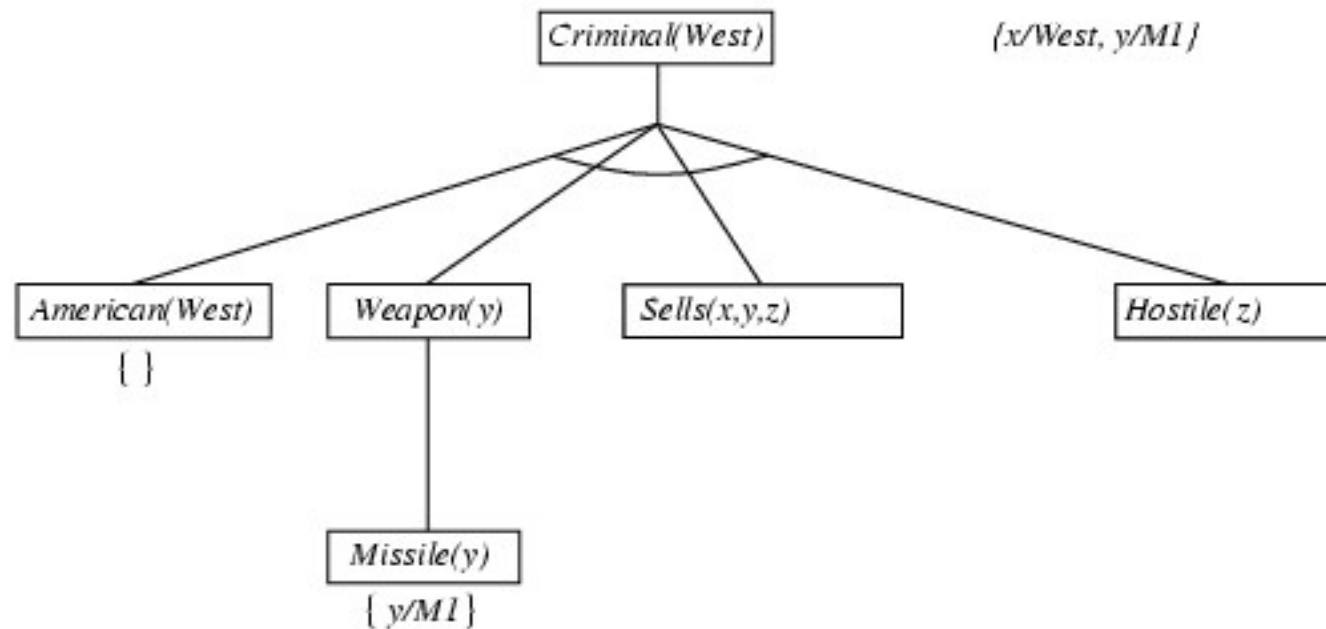
Backward Chaining Example



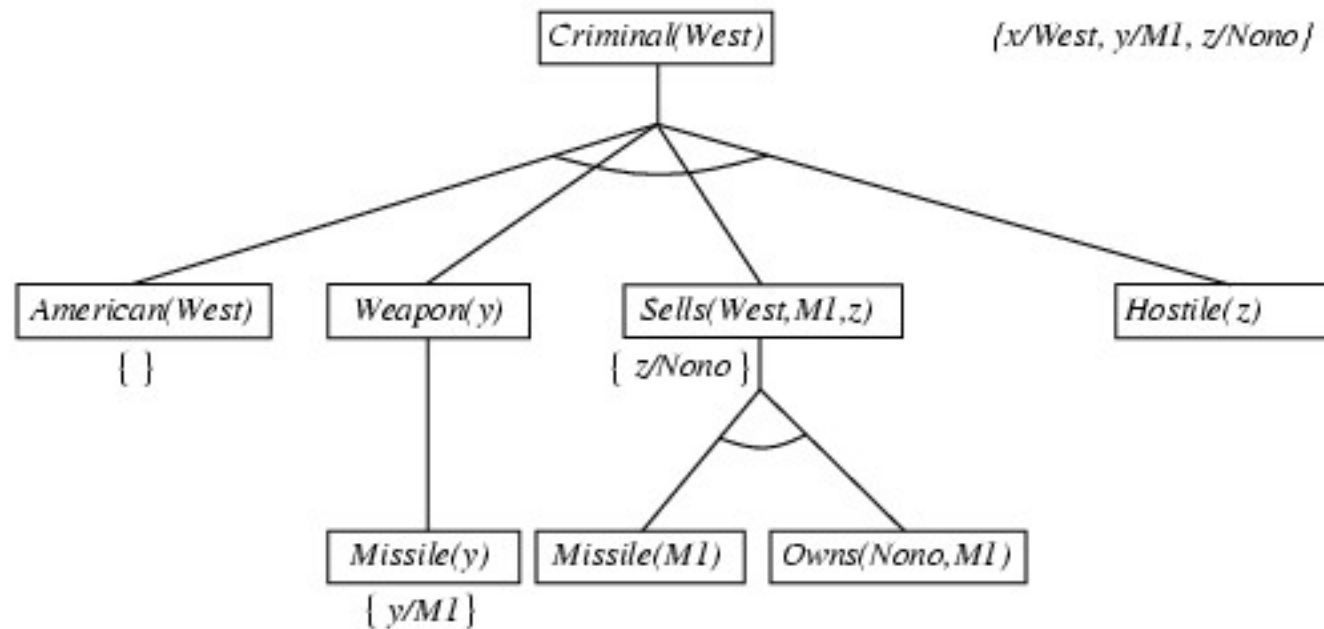
Backward Chaining Example



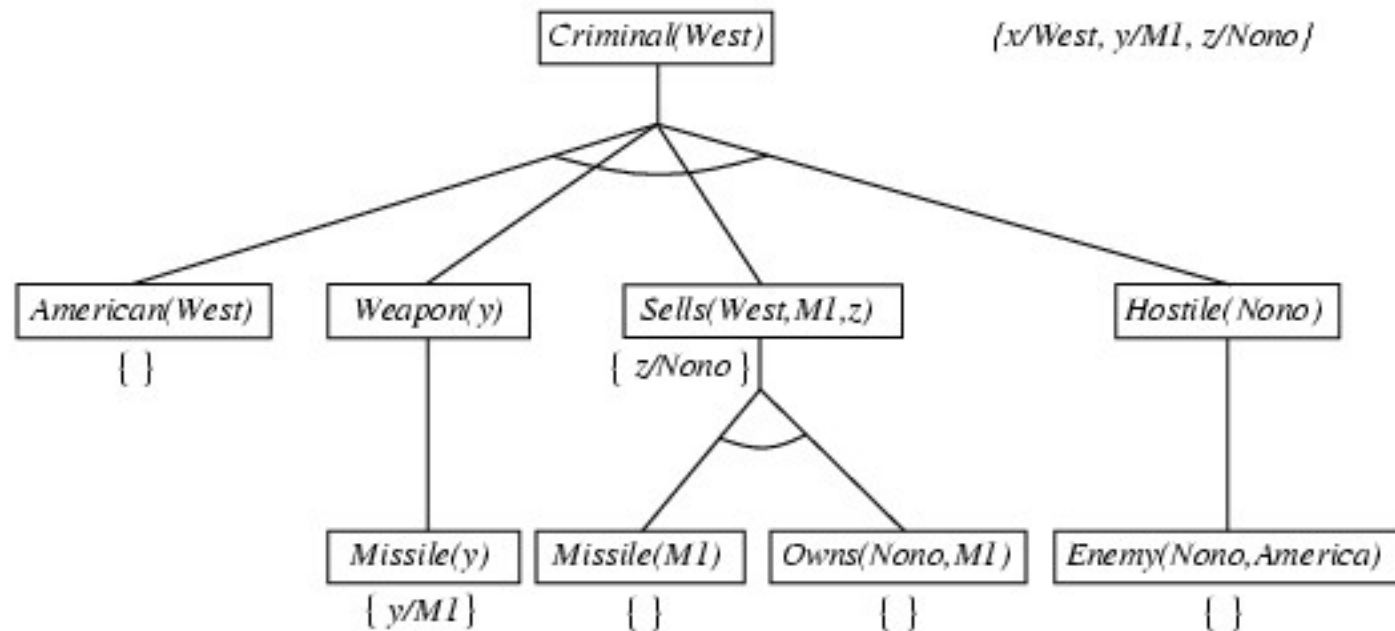
Backward Chaining Example



Backward Chaining Example



Backward Chaining Example



Analysis of Backward Chaining

- Depth-first search: space is linear in size of proof
- Repeated states and incompleteness
 - can be fixed by checking current goal against every goal on stack
- Inefficient due to repeated subgoals
 - can be fixed by caching previous results
- Widely used in logic programming

Logic Programming

- Prolog
- Lisp
- Introduced in CS 352 (symbolic programming)
- <http://www.cs.toronto.edu/~sheila/384/w11/simple-prolog-examples.html>

Exercise

- Write down logical representations for the following sentences, suitable for use with Generalized Modus Ponens
 - Horses, cows, and pigs are mammals.
 - An offspring of a horse is a horse.
 - Bluebeard is a horse.
 - Bluebeard is Charlie's parent.
 - Offspring and parent are inverse relations.

- Draw the proof tree generated by exhaustive backward-chaining algorithm for the query $\exists h \text{ Horse}(h)$, where clauses are matched in the order given.