

Class Notes CIS 675
Knowledge and Reasoning

Knowledge and Reasoning

- Agents That Reason Logically (Chapter 6)
- First-Order Logic (Chapter 7)
- Inference in First-Order Logic (Chapter 9)

I. Agents That Reason Logically

- A Knowledge-Based Agent
- Representation, Reasoning, and Logic
- Propositional Logic

A Knowledge-Based Agent

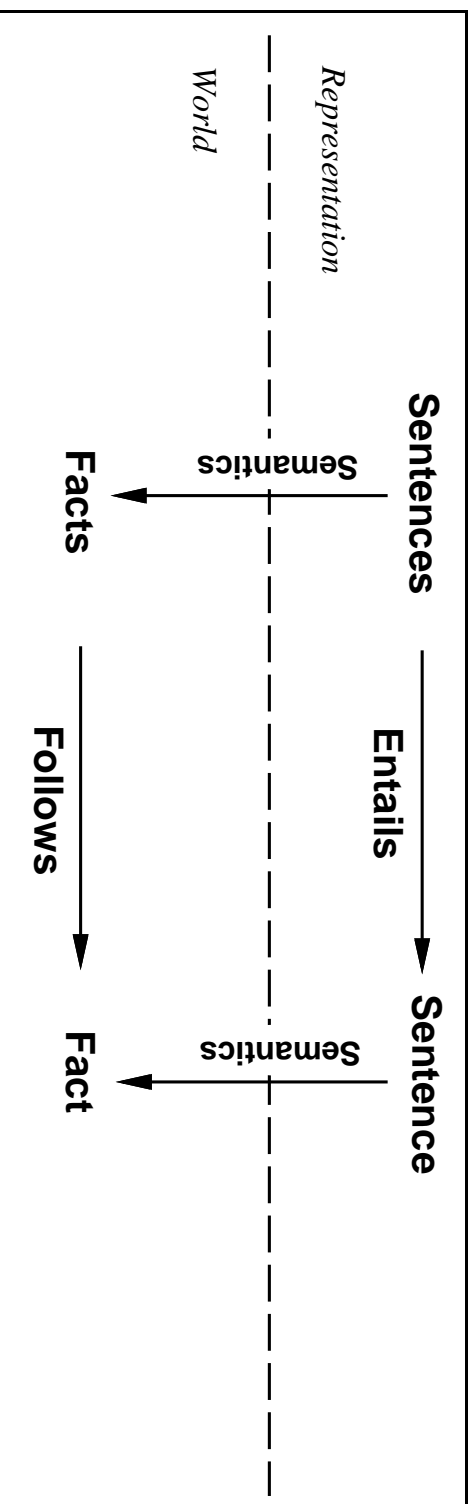
```
function KB-AGENT(percept) returns an action
  static: KB, a knowledge base
          t, a counter, initially 0, indicating time
  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  action ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```

- **Knowledge base (*KB*):** set of representations of facts. Each individual representation id called a **sentence** of a **knowledge representation language**.
Add new sentences, query knowledge: functions ASK and TELL.
- **Inference mechanism:** determine what follows from the sentences that have been added to the *KB*.

A Knowledge-Based Agent (cont'd)

- Description of knowledge-based agents in three levels:
 1. **knowledge level** or **epistemological level**: most abstract, describe agent by saying what it knows. E.g. “An automated taxi knows that the Golden Gate Bridge links San Francisco with Marin County”.
 2. **logical level**: knowledge is encoded into (formal) sentences. E.g. Links (GGBridge, SF, Marin)
 3. **implementation level**: level that runs on the agent architecture; physical representation of sentences on logical level. E.g. string, objects with pointers, or entry in 3-dim. table indexed by road links and location pairs.
- Representation of knowledge in sentences simplifies the design → **declarative** approach to system building.
- Implementation of **learning** mechanisms that produce general knowledge given a series of percepts.

Representation, Reasoning, and Logic



- **Syntax:** of a language describes the possible configurations that can constitute a sentence. Each sentence is implemented by a physical configuration or physical property of some part of the agent.
- **Semantics:** determines the facts in the world to which the sentence refer. Agents **believes** a sentence if the corresponding configuration exists.

Representation, Reasoning, and Logic (cont'd)

- Inference procedure $KB \models_i \alpha$

1. produce α for given KB
2. report whether or not α is entailed by KB .

An inference procedure that generates only entailed sentences is called **sound** or **truth-preserving**.

- **Proof Theory** of a language: reasoning steps that are sound.
- **Logics** consist of the following:
 1. A formal system for describing states of affairs, consisting of
 - (a) the **syntax** of the language (how to make sentences), and
 - (b) the **semantics** of the language (how sentence relates to states of affairs).
 2. The **proof-theory** \rightarrow a set of rules for deducing the entailments of a set of sentences.

Representation, Reasoning, and Logic (cont'd)

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, times	true/false/unknown
Probability theory	facts	degree of belief 0...1
Fuzzy logic	degree of truth	degree of belief 0...1

Ontology, branch of metaphysics that deals with the nature of being.

Epistemology, the theory of knowledge, esp. the critical study of its validity, methods, and scope.

Propositional Logic

- Symbols: logical constants *True* and *False*, propositional symbols *P* and *Q*, logical connectives \wedge , \vee , \Leftrightarrow , \Rightarrow , \neg , parentheses, ().
- Grammar in BNF (Backus-Naur Form)

Sentence \rightarrow *AtomicSentence* | *ComplexSentence*

AtomicSentence \rightarrow **True** | **False**

ComplexSentence \rightarrow (*Sentence*)

| *Sentence* *Connective* *Sentence*

| \neg *Sentence*

Connective \rightarrow \wedge | \vee | \Leftrightarrow | \Rightarrow

Propositional Logic (cont'd)

Semantics

- Proposition symbols can carry any meaning, interpretation is an arbitrary fact.
- Logical constants: *True* means the fact is always true, under any interpretation, *False* means the fact is false.
- Connectives can be defined by a truth table:

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>	<i>False</i>	<i>True</i>	<i>True</i>
<i>False</i>	<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>False</i>
<i>True</i>	<i>False</i>	<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>	<i>False</i>
<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i>

Propositional Logic (cont'd)

Inference Rules

- **Modus Ponens:** from an implication and the premise of the implication, you can infer the conclusion.

$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$$

- **And-Elimination:** from a conjunction, you can infer any of the conjuncts.

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}{\alpha_i}$$

- **And-Introduction:** from a sequence of sentences, you can infer their conjunctions.

$$\frac{\alpha_1, \alpha_2, \dots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}$$

Propositional Logic (cont'd)

More Inference Rules

- **Or-Introduction:** from a sentence, you can infer its disjunction with anything.

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \dots \vee \alpha_n}$$

- **Double-Negation Elimination:**

$$\frac{\neg\neg\alpha}{\alpha}$$

- **Unit Resolution:**

$$\frac{\alpha \vee \beta, \neg\beta}{\alpha}$$

- **Resolution:**

$$\frac{\alpha \vee \beta, \neg\beta \vee \gamma}{\alpha \vee \gamma} \text{ or equivalently } \frac{\alpha \Rightarrow \beta, \neg\beta \Rightarrow \gamma}{\alpha \Rightarrow \gamma}$$

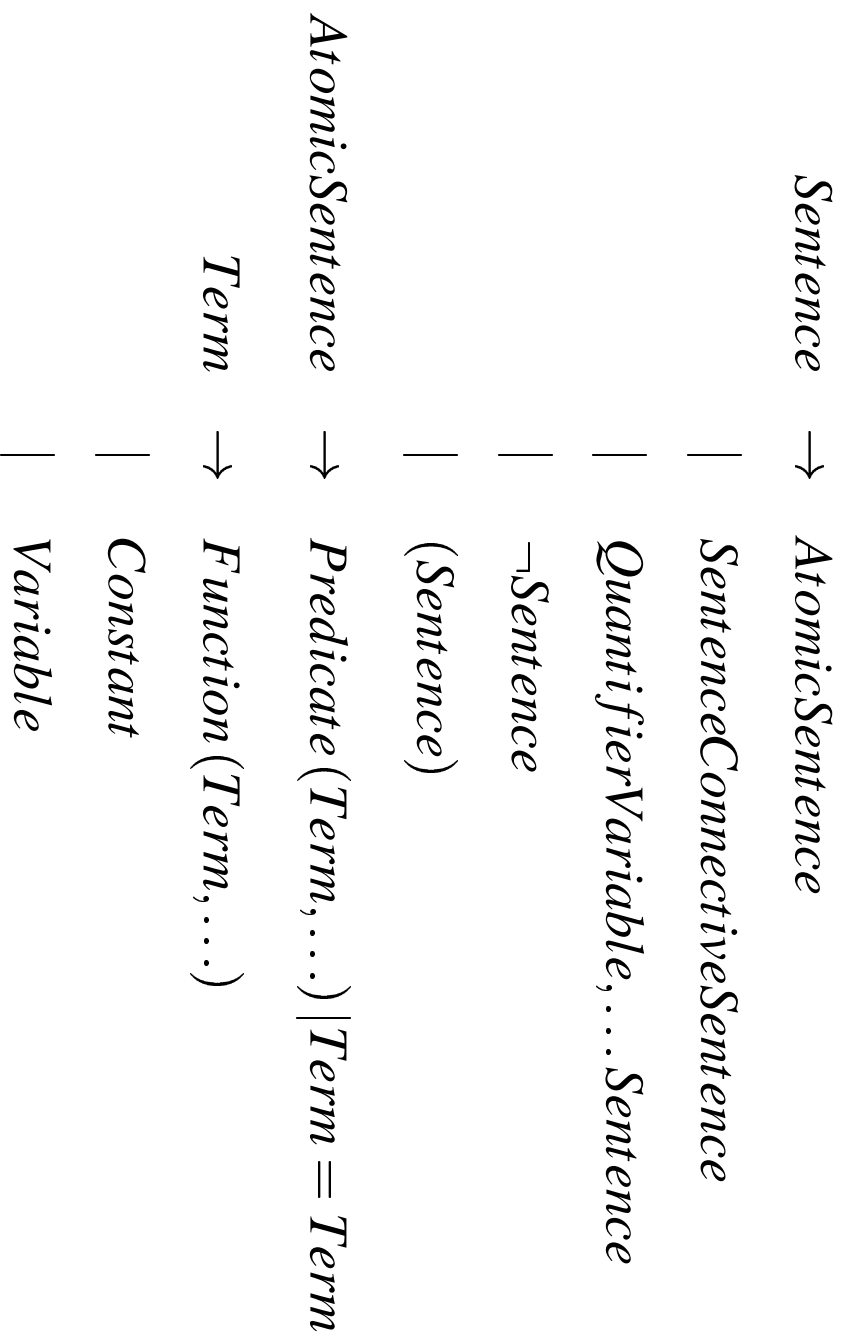
II. First-Order Logic

- Ontological commitment: world consists of **objects**, i.e., things with individual identities and **properties** that distinguish them from other objects. Among the objects, various **relations** hold. Some of the relations are **functions** – relations in which there is only one value for a given input.
- Examples:
 - “One plus two equals three” Objects: one, two, three, one plus two; Relations: equals; Function: plus.
 - “Squares neighboring the wumpus are smelly” Objects: wumpus, square; Property: smelly; Relation: neighboring.
- also called *First-Order Predicate Calculus*.

First-Order Logic

- Syntax and Semantics
- (Extension and Notational Variations)
- Using First-Order Logic
- Prolog Examples*

Syntax and Semantics



Syntax and Semantics (cont'd)

Connective $\rightarrow \Rightarrow \mid \wedge \mid \vee \mid \Leftrightarrow$

Quantifier $\rightarrow \forall \mid \exists$

Constant $\rightarrow a \mid x_1 \mid john \mid \dots$

Variable $\rightarrow A \mid X \mid Name \mid \dots$

Predicate $\rightarrow before \mid hasColor \mid raining \mid \dots$

Function $\rightarrow mother \mid leftLegOf \mid \dots$

Syntax and Semantics (cont'd)

- **Term:** logical expression that refers to an object.
- **Atomic sentence:** terms \rightarrow objects, predicate symbols \rightarrow relations \implies state facts.

brother(richard, john)

married(fatherOf(richard), motherOf(john))

- **Complex Sentence:** using logical connectives.
- **Quantifiers:** allows to express properties of entire collections of objects, rather than having to enumerate the objects by name.

Syntax and Semantics: Quantifiers

- Substituting the name *john* of an object for a variable *PERSON*, binding variables $\{john/PERSON\}$.

- **Universal quantification** (\forall)

$$\forall ANIMAL \ cat(ANIMAL) \Rightarrow mammal(ANIMAL)$$

“For all animals that are cats we can imply they are mammals” is true for all possible values of *ANIMAL*.

- **Existential quantification** (\exists)

$$\exists ANIMAL \ sister(ANIMAL, spot) \wedge cat(ANIMAL)$$

“There exists an animal that is the sister of Spot and it is also a cat” is true for at least one value.

Using First-Order Logic

- **Domain:** a section of the world about which we wish to express some knowledge; use of **axioms**, and **definitions** to capture basic facts of the domain.

- Example:

$$\forall M, C \text{ mother}(M, C) \Leftrightarrow \text{female}(M) \wedge \text{parent}(M, C)$$

$$\forall W, H \text{ husband}(H, W) \Leftrightarrow \text{male}(H) \wedge \text{spouse}(H, W)$$

$$\forall X \text{ male}(X) \Leftrightarrow \neg \text{female}(X)$$

$$\forall P, C \text{ parent}(P, C) \Leftrightarrow \text{child}(C, P)$$

$$\forall G, C \text{ grandparent}(G, C) \Leftrightarrow \exists P \text{ parent}(G, P) \wedge \text{parent}(P, C)$$

$$\forall X, Y \text{ sibling}(X, Y) \Leftrightarrow X \neq Y \wedge \exists P \text{ parent}(P, X) \wedge \text{parent}(P, Y)$$

Prolog Examples*

- Syntax: similar to first-order logic (predicate calculus).
- Exceptions:

$$\alpha \Rightarrow \beta \quad \longrightarrow \quad \text{beta} \text{ :- alpha.}$$

$$\alpha \wedge \beta \quad \longrightarrow \quad \text{alpha, beta.}$$

$$\alpha \vee \beta \quad \longrightarrow \quad \text{alpha; beta | alpha. beta.}$$

- No equivalence \Leftrightarrow .

III. Inference in First-Order Logic

- Inference Rules Involving Quantifiers
- Generalized Modus Ponens
- Forward and Backward Chaining
- Completeness and Resolution

Inference Rules Involving Quantifiers

Notation

$\text{SUBST}(\{x/Sam, y/Pam\}, Likes(x, y)) = Likes(Sam, Pam)$

- **Universal Elimination:** For any sentence α , variable v , and ground

term^a g :

$$\frac{\forall v \alpha}{\text{SUBST}(\{v, g\}, \alpha)}$$

For example, from $\forall x Likes(x, IceCream)$, we can use substitution $\{x/Ben\}$ and infer $Likes(Ben, IceCream)$.

^aground term does not include variables

- **Existential Elimination:** For any sentence α , variable v , and constant symbol k that does not appear elsewhere in the knowledge base:

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

For example: from $\exists x \textit{Kill}(x, \textit{Victim})$, we can infer $\textit{Kill}(\textit{Murderer}, \textit{Victim})$, as long as *Murderer* does not appear elsewhere.

- **Existential Introduction:** For any sentence α , variable v that does not occur in α , and ground term g that does occur in α :

$$\frac{\alpha}{\exists v \text{SUBST}(\{g/v\}, \alpha)}$$

For example: from $\textit{Likes}(\textit{Jerry}, \textit{IceCream})$ we can infer $\exists x \textit{Likes}(x, \textit{IceCream})$.

Using Inference Rules

“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.”

$$\forall x, y, z \text{ American}(x) \wedge \text{Weapon}(y) \wedge \text{Nation}(z) \wedge \text{Hostile}(z) \quad (1)$$

$$\wedge \text{Sells}(x, y, z) \Rightarrow \text{Criminal}(x)$$

$$\exists x \text{ Owns}(\text{Nono}, x) \wedge \text{Missile}(x) \quad (2)$$

$$\forall x \text{ Owns}(\text{Nono}, x) \wedge \text{Missile}(x) \Rightarrow \text{Sells}(\text{West}, \text{Nono}, x) \quad (3)$$

$$\forall x \text{ Missile}(x) \Rightarrow \text{Weapon}(x) \quad (4)$$

$$\forall x \text{ Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x) \quad (5)$$

$$\text{American}(\text{West}) \quad (6)$$

$$\text{Nation}(\text{Nono}) \quad (7)$$

$$\text{Enemy}(\text{Nono}, \text{America}) \quad (8)$$

$$\text{Nation}(\text{America}) \quad (9)$$

Using Inference Rules (cont'd)

- Process of finding proof can be formulated as a search process: Initial state = KB (sentences 1–9)
Operators = applicable inference rules
Goal Test = KB containing *Criminal(West)*
- Important characteristics
 - The proof is 14 steps long.
 - The branching factor increases as the knowledge base grows (because some inference rules combine existing facts).
 - Universal Elimination can have an enormous branching factor (because we can replace the variable by any ground term).
 - Most common step: combining atomic sentences into conjunctions, instantiating universal rules to match, and then apply Modus Ponens.

Generalized Modus Ponens

- Combine And-Introduction, Universal Elimination, and Modus

Ponens:

Missile(*M1*)

Owns(*Nono*, *M1*)

$\forall x \text{ Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, \text{Nono}, x)$

infer in one step

Sells(*West*, *Nono*, *M1*)

- Find some *x* in the knowledge base such that *x* is a missile and *Nono* owns *x*, and then infer that *West* sells the missile to *Nono*.
- Even if we knew $\forall y \text{ Owns}(y, M1)$ (more general assertion), we can find the substitutions $\{x/M1\}$ and $\{y/Nono\}$.

Generalized Modus Ponens (cont'd)

- **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 *Missile*($M1$), p'_2 is *Owms*($y, M1$), θ is $\{x/M1, y/Nono\}$, p_1 is *Missile*(x), p_2 is *Owms*(*Nono*, x), q is *Sells*(*West*, *Nono*, x), and
 $\text{SUBST}(\theta, q)$ is *Sells*(*West*, *Nono*, $M1$).

Generalized Modus Ponens – Canonical Form

- Attempt to build an inference mechanism with one inference rule \rightarrow Generalized Modus Ponens; a sentence is either an atomic sentence or an implication with a conjunction of atomic sentences on the left (premise) and a single atomic sentence on the right. (**Horn sentences, Horn Normal Form**)
- One can convert sentences into Horn sentences when they are entered into the KB, using Existential Elimination and And-Elimination: e.g. $\exists x \text{ Owns}(\text{Nono}, x) \wedge \text{Missile}(x)$ is converted into two atomic Horn sentences ($\text{Owns}(\text{Nono}, M1)$ and $\text{Missile}(M1)$).
- The universal quantifier is dropped since the existential quantifier is eliminated.

Generalized Modus Ponens – Unification

- The unification routine UNIFY takes two atomic sentences p and q and returns a substitution that would make p and q look the same. If there is no such substitution, then UNIFY returns *fail*.

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

θ is called the **unifier** of the two sentences.

Unification Example

- Assertion:

$Knows(John, x) \Rightarrow Hates(John, x)$

Use Modus Ponens to find out who he hates.

- Knowledge base:

$Knows(John, Jane)$

$Knows(y, Leon)$

$Knows(y, Mother(y))$

$Knows(x, Elisabeth)$

x and y are implicitly universally quantified.

- Unification

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

$$\text{UNIFY}(\text{Knows}(\text{John}, x), \text{Knows}(y, \text{Leon})) = \{x/\text{Leon}, y/\text{John}\}$$

$$\text{UNIFY}(\text{Knows}(\text{John}, x), \text{Knows}(y, \text{Mother}(y))) = \{y/\text{John}, x/\text{Mother}(\text{John})\}$$

$$\text{UNIFY}(\text{Knows}(\text{John}, x), \text{Knows}(x, \text{Elisabeth})) = \text{fail}$$

- The last unification fails because x cannot take the value *John* and *Elisabeth* at the same time. A way to solve this problem is to **standardize apart** the two sentences being unified: rename variables to avoid the conflict.

- **Most general unifier** makes the least commitment about binding of variables.

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

OR $\{y/\text{John}, x/z/, w/\text{Fritz}\}$ etc.

Proof: West is a criminal.

Forward and Backward Chaining

```
procedure FORWARD-CHAIN(KB, p)  
  if there is a sentence in KB that is a renaming of p then return  
  Add p to KB  
  for each ( $p_1 \wedge \dots \wedge p_n \Rightarrow q$ ) in KB such that for some i, UNIFY( $p_i, p$ ) =  $\theta$  succeeds do  
    FIND-AND-INFER(KB, [ $p_1, \dots, p_{i-1}, p_{i+1}, \dots, p_n$ ],  $q, \theta$ )  
  end  
  
  procedure FIND-AND-INFER(KB, premises, conclusion,  $\theta$ )  
    if premises = [] then  
      FORWARD-CHAIN(KB, SUBST( $\theta$ , conclusion))  
    else for each p' in KB such that UNIFY( $p',$  SUBST( $\theta$ , FIRST(premises))) =  $\theta_2$  do  
      FIND-AND-INFER(KB, REST(premises), conclusion, COMPOSE( $\theta, \theta_2$ ))  
    end
```

Forward and Backward Chaining (cont'd)

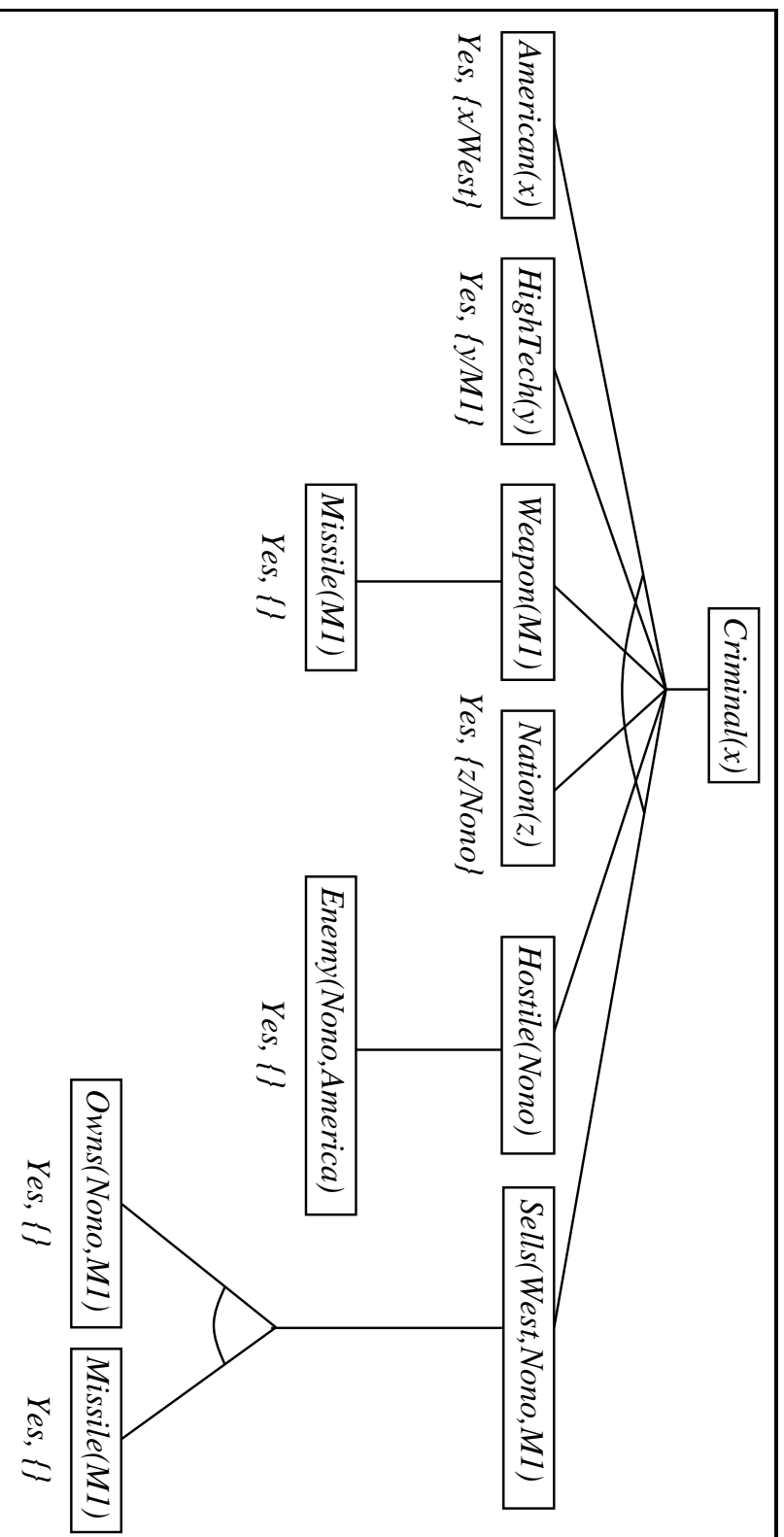
```
function BACK-CHAIN(KB, q) returns a set of substitutions
BACK-CHAIN-LIST(KB, [q], {})



---

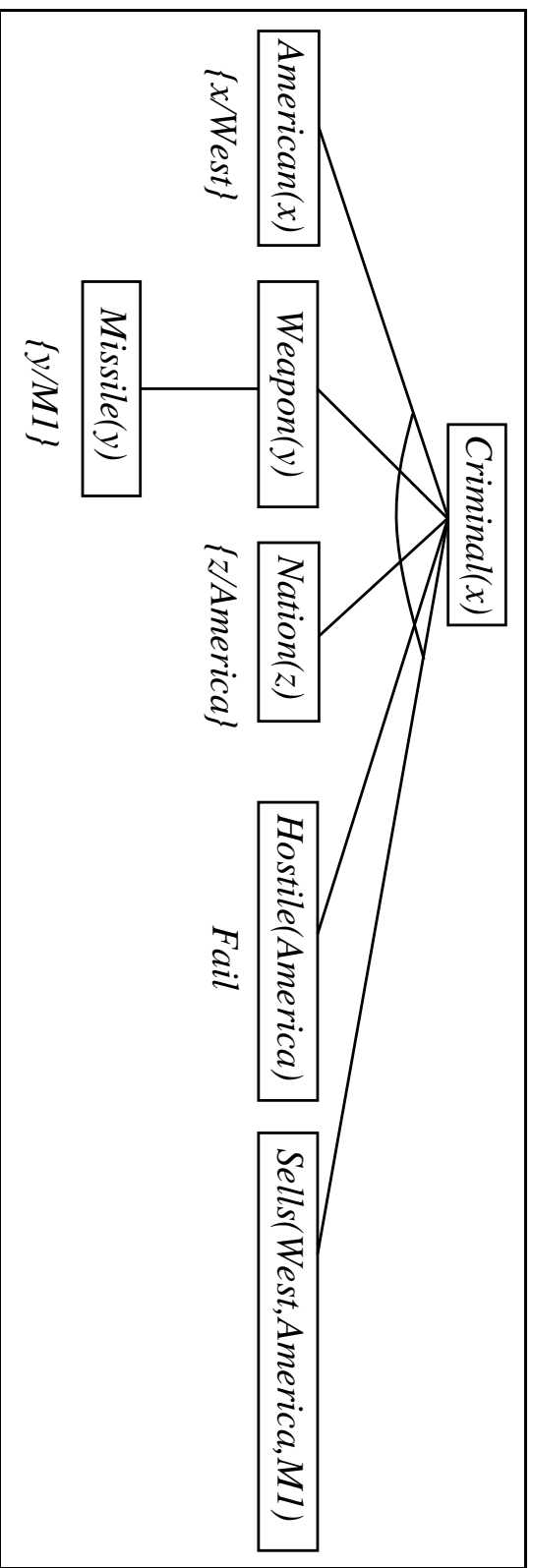

function BACK-CHAIN-LIST(KB, qlist,  $\theta$ ) returns a set of substitutions
inputs: KB, a knowledge base
         qlist, a list of conjuncts forming a query ( $\theta$  already applied)
          $\theta$ , the current substitution
static: answers, a set of substitutions, initially empty
if qlist is empty then return { $\theta$ }
          $q \leftarrow \text{FIRST}(qlist)$ 
         for each  $q_i$  in KB such that  $\theta_i \leftarrow \text{UNIFY}(q, q_i)$  succeeds do
             Add COMPOSE( $\theta, \theta_i$ ) to answers
         end
         for each sentence ( $p_1 \wedge \dots \wedge p_n \Rightarrow q_i$ ) in KB such that  $\theta_i \leftarrow \text{UNIFY}(q, q_i)$  succeeds do
             answers  $\leftarrow$  BACK-CHAIN-LIST(KB, SUBST( $\theta_i$ , [ $p_1 \dots p_n$ ]), COMPOSE( $\theta, \theta_i$ ))  $\cup$  answers
         end
return the union of BACK-CHAIN-LIST(KB, REST(qlist),  $\theta$ ) for each  $\theta \in$  answers
```

Forward and Backward Chaining (cont'd)



Proof tree to infer that West is a criminal. Backward-chaining: start with goal at the root; forward-chaining: add knowledge as leaves, and combine.

Forward and Backward Chaining (cont'd)



If the wrong substitution is made the inference process fails. Backtrack!

Completeness

- Given the knowledge base

$$\forall x \ P(x) \Rightarrow Q(x)$$

$$\forall x \ \neg P(x) \Rightarrow R(x)$$

$$\forall x \ Q(x) \Rightarrow S(x)$$

$$\forall x \ R(x) \Rightarrow S(x)$$

one should conclude $S(A)$; $S(A)$ is true if $Q(A)$ or $R(A)$ are true, one of those must be true because either $P(A)$ or $\neg P(A)$ is true.

- Chaining with Modus Ponens cannot derive $S(A)$, because $\forall x \ \neg P(x) \Rightarrow R(x)$ cannot be converted to Horn form.
- The proof procedure using Modus Ponens is **incomplete**. A complete proof procedure for first-order logic would enable a machine to solve any problem that can be stated in that language.

Completeness – Remedy

- Kurt Gödel showed that for first-order logic, any sentence that is entailed by another set of sentences can be proved from the set. (1930–31). The **completeness theorem** says there is a solution, but doesn't say how to get it.
- J. A. Robinson published **resolution algorithm** (1965).

$$\frac{\alpha \vee \beta, \neg \beta \vee \gamma}{\alpha \vee \gamma} \quad \text{or equivalently} \quad \frac{\neg \alpha \Rightarrow \beta, \beta \Rightarrow \gamma}{\neg \alpha \Rightarrow \gamma}$$