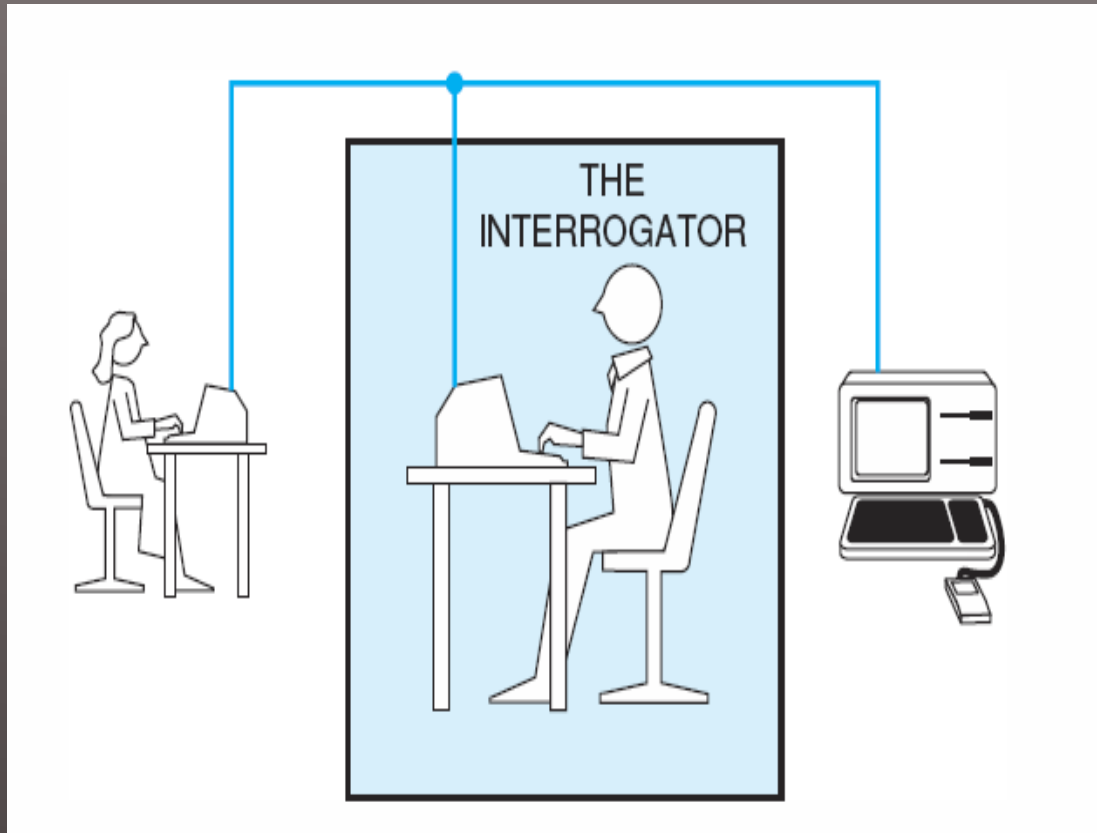


Artificial Intelligence

DEFINITION

- ▶ *AI — the branch of computer science that is concerned with the automation of intelligent behavior*
 - ▶ *Sound theoretical and applied principles*
 - ▶ *Data structures for knowledge representation*
 - ▶ *Algorithms of applying knowledge*
 - ▶ *Languages for algorithm implementation*
- ▶ Problem
 - ▶ What is Intelligence?

THE TURING TEST



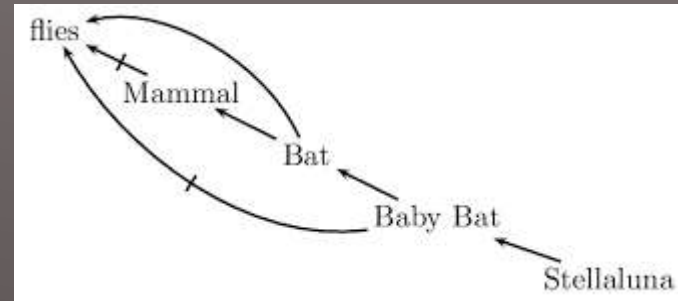
The interrogator

- cannot see and speak to either
- does not know which is actually machine
- May communicate with them solely by textual device

If the interrogator cannot distinguish the machine from the human, then the machine may be assumed to be intelligent.

MODELS OF INTELLIGENCE

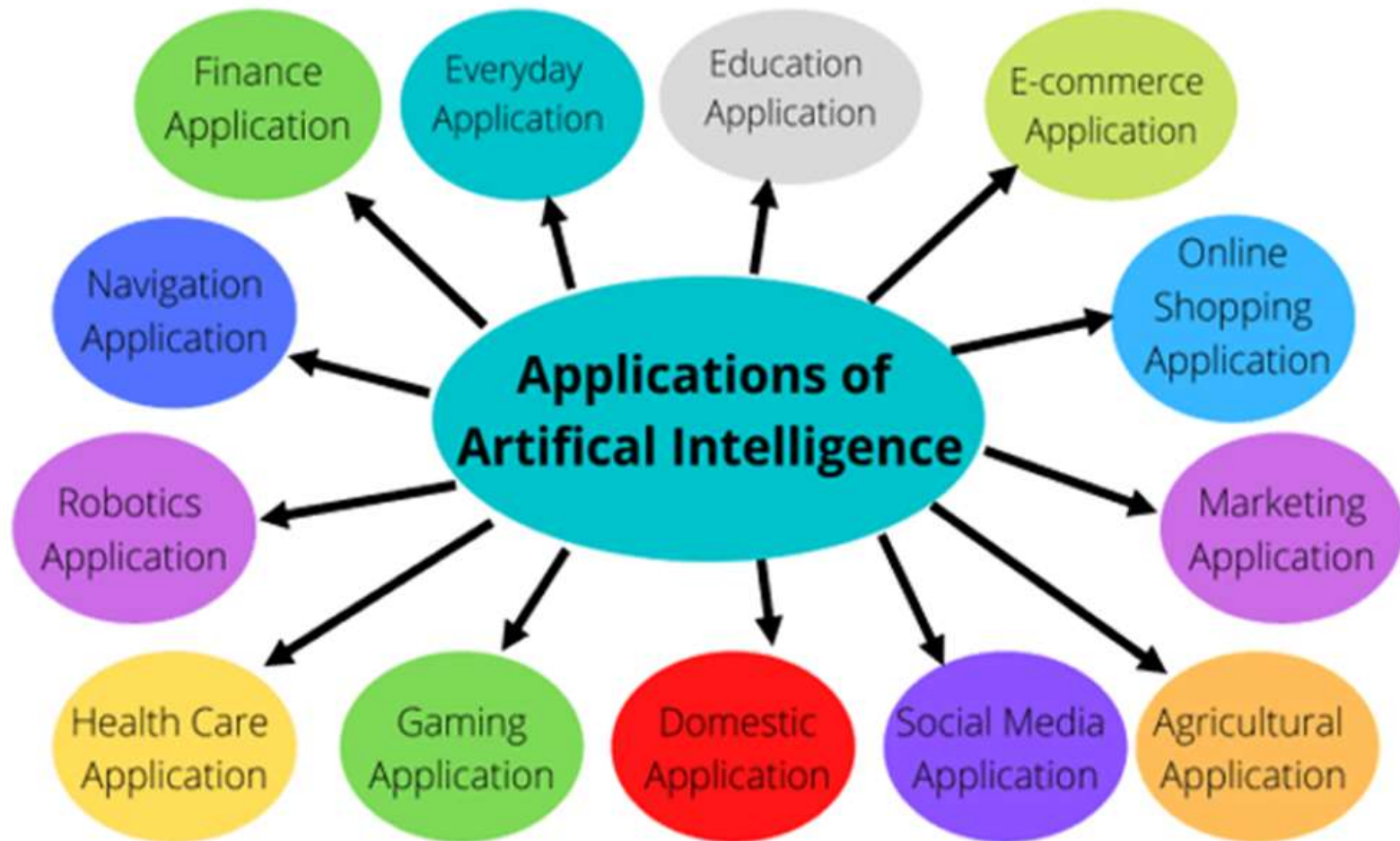
- ▶ Logic Models
 - ▶ Formal logic Truth table
 - ▶ Probability logic - Bayesian
 - ▶ Fuzzy logic – Multiple true conditions
 - ▶ Non-monotonic logic -
- ▶ Biological Models
 - ▶ Connectionist model
 - ▶ Genetic algorithms
 - ▶ Evolution
- ▶ Social Models
 - ▶ Network models
 - ▶ Behaviors
- ▶ Agents -- Combination



AGENTS THEORIES

- ▶ Intelligence is reflected by a collective behaviors of large numbers of very simple interacting, semi-autonomous individuals – agents
- ▶ Various agents
 - ▶ Rote agents
 - ▶ Coordination agents
 - ▶ Search agents
 - ▶ Learning agents
 - ▶ Decision agents
- ▶ Agents design
 - ▶ Structure for information representation
 - ▶ Search strategies
 - ▶ Architecture for supporting the interaction of agents

AI RESEARCH AND APPLICATION AREAS



Important Features of Artificial Intelligence

1. Reasoning, pattern recognition, learning - Inference.
2. Heuristic search for non-algorithmic solutions.
3. Problem-solving using inexact, missing, or poorly defined information using representational formalisms
4. Reasoning - Qualitative features of a situation.

Important Features of Artificial Intelligence

5. Deals with semantic meaning as well as syntactic form.
6. Obtaining answers that are neither exact nor optimal, but sufficient
7. Expert systems - Solving problems using large amounts of domain-specific knowledge
8. Problem-solving with meta-level knowledge to control strategies

Artificial Intelligence as Representation and Search

REPRESENTATION SYSTEMS

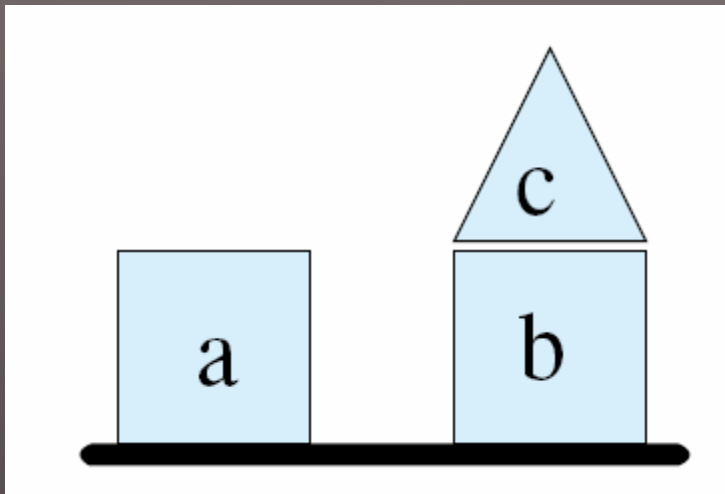
- ▶ What is it?
 - ▶ *Capture the essential features of a problem domain and make that information accessible to a problem-solving procedure*
- ▶ Measures to evaluate
 - ▶ Abstraction – how to manage complexity
 - ▶ Expressiveness – what can be represented
 - ▶ Efficiency – how is it used to solve problems
- ▶ Trade-off between efficiency and expressiveness

Features of AI Representation Language:

- Handle qualitative Knowledge
- Allow new knowledge to be inferred from a set of facts and rules
- Allow representation of general principles as well as specific situations
- Capture complex semantic meaning
- Allow meta-level reasoning

Block World Representation

A blocks world



General rule

Logical Clauses describing some important properties and relationships

```
clear(c)
clear(a)
ontable(a)
ontable(b)
on(c, b)
cube(b)
cube(a)
pyramid(c)
```

$$\forall X \neg \exists Y \text{ on}(Y,X) \Rightarrow \text{clear}(X)$$

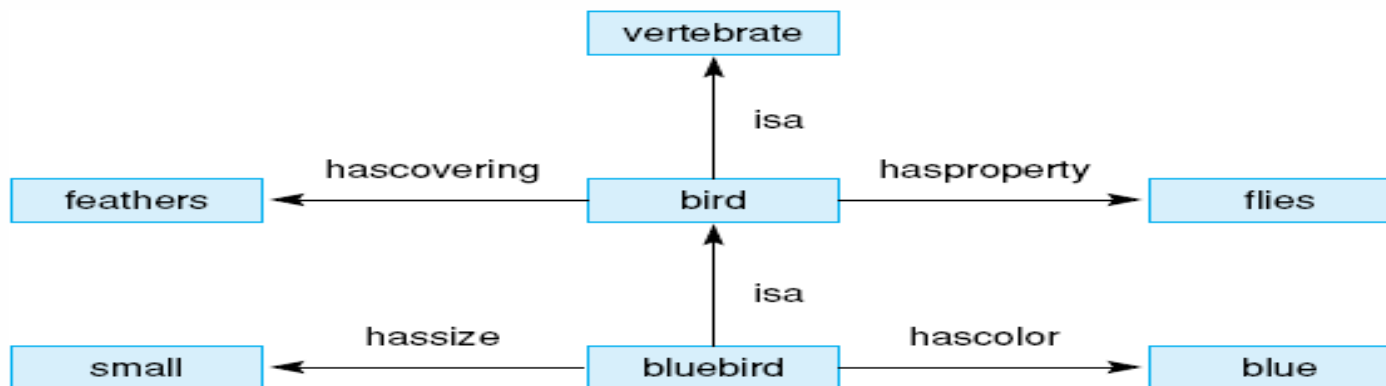
Bluebird Representations

A bluebird is a small blue-colored bird and a bird is a feathered flying vertebrate.

Logical predicates representing a simple description of a bluebird.

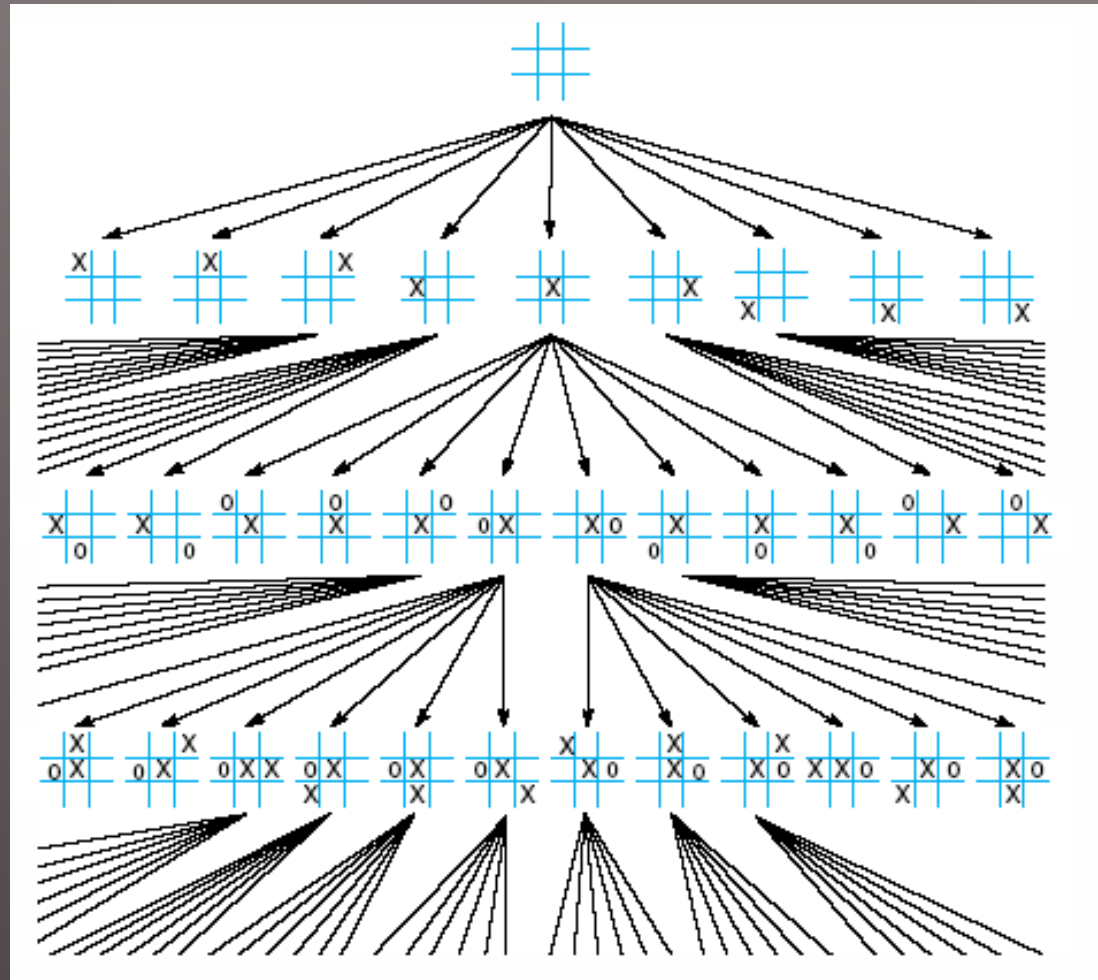
```
hassize(bluebird,small)
hascovering(bird,feathers)
hascolor(bluebird,blue)
hasproperty(bird,flies)
isa(bluebird,bird)
isa(bird,vertebrate)
```

Semantic network description of a bluebird.



Tic-tac-toe State Space

Portion of the
state space for
tic-tac-toe.

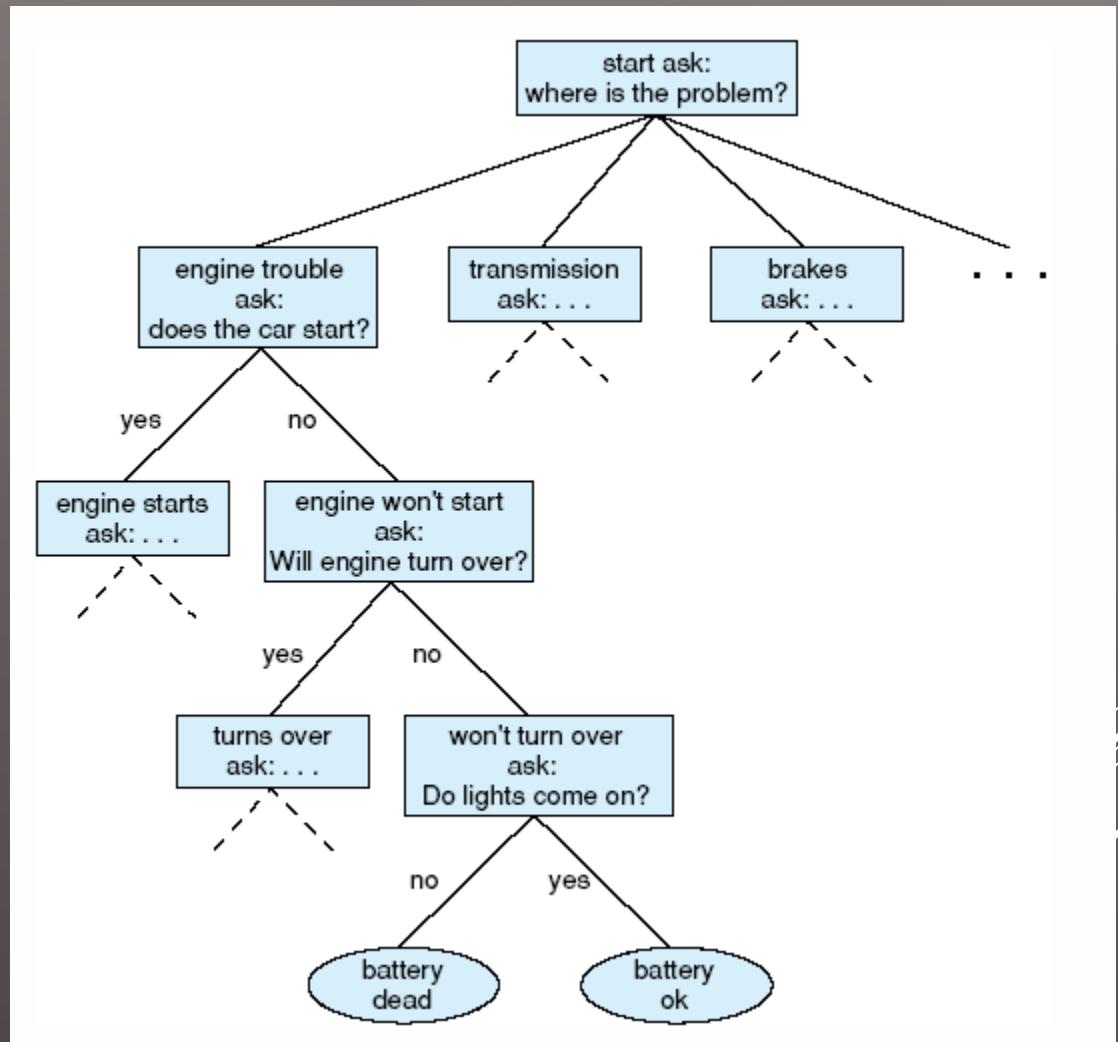


STATE SPACE SEARCH- PROBLEM SOLVING

- ▶ State space
 - ▶ State – any current representation of a problem
 - ▶ State space
 - ▶ All possible state of the problem
 - ▶ Start states – the initial state of the problem
 - ▶ Target states – the final states of the problem that has been solved
 - ▶ State space graph
 - ▶ Nodes – possible states
 - ▶ Links – actions that change the problem from one state to another
- ▶ State space search
 - ▶ Find a path from an initial state to a target state in the state space
 - ▶ Various search strategies
 - ▶ Exhaustive search – guarantee that the path will be found if it exists
 - ▶ Depth-first
 - ▶ Breath-first
 - ▶ Best-first search
 - ▶ heuristics

Auto Diagnosis State Space

State space
description of
the automotive
diagnosis
problem.



2

The Predicate Calculus

2.0 Introduction

2.1 The Propositional Calculus

2.2 The Predicate Calculus

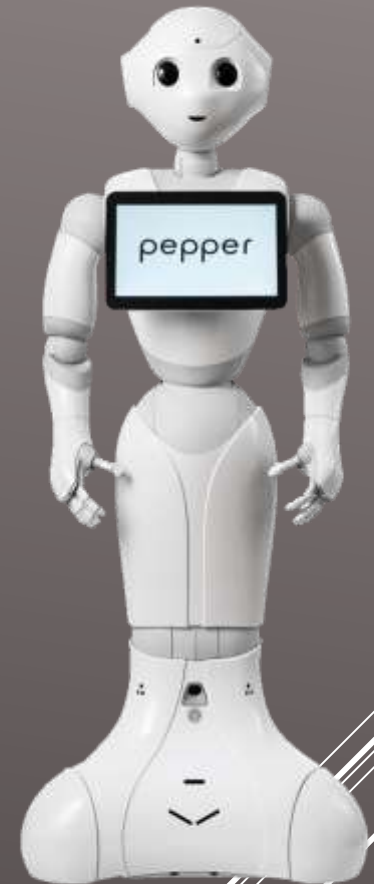
2.3 Using Inference Rules to Produce Predicate Calculus Expressions

2.4 Application: A Logic-Based Financial Advisor

Knowledge bases



- Knowledge base = set of **sentences** in a **formal** language
- **Declarative** approach to building an agent (or other system):
 - **Tell** it what it needs to know
- Then it can **Ask** itself what to do - answers should follow from the KB
- Agents can be viewed at the **knowledge level**
i.e., what they know, regardless of how implemented
- Or at the **implementation level**
 - i.e., data structures in KB and algorithms that manipulate them



A simple 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
```

The agent must be able to:

- Represent states, actions, etc.
- Incorporate new percepts
- Update internal representations of the world
- Deduce hidden properties of the world
- Deduce appropriate actions

Logic in general

- **Logics** are formal languages for representing information such that conclusions can be drawn
- **Syntax** defines the sentences in the language
- **Semantics** define the "meaning" of sentences;
 - i.e., define **truth** of a sentence in a world
- E.g., the language of arithmetic
 - $x+2 \geq y$ is a sentence; $x^2+y > \{ \}$ is not a sentence

DEFINITION

PROPOSITIONAL CALCULUS SYMBOLS

The *symbols* of propositional calculus are the propositional symbols:

P, Q, R, S, ...

truth symbols:

true, false

and connectives:

$\wedge, \vee, \neg, \rightarrow, \equiv$

DEFINITION

PROPOSITIONAL CALCULUS SENTENCES

Every propositional symbol and truth symbol is a sentence.

For example: **true**, **P**, **Q**, and **R** are sentences.

The *negation* of a sentence is a sentence.

For example: $\neg \mathbf{P}$ and $\neg \mathbf{false}$ are sentences.

The *conjunction*, or *and*, of two sentences is a sentence.

For example: $\mathbf{P} \wedge \neg \mathbf{P}$ is a sentence.

The *disjunction*, or *or*, of two sentences is a sentence.

For example: $\mathbf{P} \vee \neg \mathbf{P}$ is a sentence.

The *implication* of one sentence from another is a sentence.

For example: $\mathbf{P} \rightarrow \mathbf{Q}$ is a sentence.

The *equivalence* of two sentences is a sentence.

For example: $\mathbf{P} \vee \mathbf{Q} \equiv \mathbf{R}$ is a sentence.

Legal sentences are also called *well-formed formulas* or *WFFs*.

DEFINITION

PROPOSITIONAL CALCULUS SEMANTICS

An *interpretation* of a set of propositions is the assignment of a truth value, either **T** or **F**, to each propositional symbol.

The symbol **true** is always assigned **T**, and the symbol **false** is assigned **F**.

The interpretation or truth value for sentences is determined by:

The truth assignment of *negation*, $\neg P$, where **P** is any propositional symbol, is **F** if the assignment to **P** is **T**, and **T** if the assignment to **P** is **F**.

The truth assignment of *conjunction*, \wedge , is **T** only when both conjuncts have truth value **T**; otherwise it is **F**.

The truth assignment of *disjunction*, \vee , is **F** only when both disjuncts have truth value **F**; otherwise it is **T**.

The truth assignment of *implication*, \rightarrow , is **F** only when the premise or symbol before the implication is **T** and the truth value of the consequent or symbol after the implication is **F**; otherwise it is **T**.

The truth assignment of *equivalence*, \equiv , is **T** only when both expressions have the same truth assignment for all possible interpretations; otherwise it is **F**.

For propositional expressions **P**, **Q** and **R**:

- Two sentences are **logically equivalent** iff true in same models: $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

$$\neg(\neg \mathbf{P}) \equiv \mathbf{P}$$

$$(\mathbf{P} \vee \mathbf{Q}) \equiv (\neg \mathbf{P} \rightarrow \mathbf{Q})$$

the contrapositive law: $(\mathbf{P} \rightarrow \mathbf{Q}) \equiv (\neg \mathbf{Q} \rightarrow \neg \mathbf{P})$

de Morgan's law: $\neg(\mathbf{P} \vee \mathbf{Q}) \equiv (\neg \mathbf{P} \wedge \neg \mathbf{Q})$ and $\neg(\mathbf{P} \wedge \mathbf{Q}) \equiv (\neg \mathbf{P} \vee \neg \mathbf{Q})$

the commutative laws: $(\mathbf{P} \wedge \mathbf{Q}) \equiv (\mathbf{Q} \wedge \mathbf{P})$ and $(\mathbf{P} \vee \mathbf{Q}) \equiv (\mathbf{Q} \vee \mathbf{P})$

the associative law: $((\mathbf{P} \wedge \mathbf{Q}) \wedge \mathbf{R}) \equiv (\mathbf{P} \wedge (\mathbf{Q} \wedge \mathbf{R}))$

the associative law: $((\mathbf{P} \vee \mathbf{Q}) \vee \mathbf{R}) \equiv (\mathbf{P} \vee (\mathbf{Q} \vee \mathbf{R}))$

the distributive law: $\mathbf{P} \vee (\mathbf{Q} \wedge \mathbf{R}) \equiv (\mathbf{P} \vee \mathbf{Q}) \wedge (\mathbf{P} \vee \mathbf{R})$

the distributive law: $\mathbf{P} \wedge (\mathbf{Q} \vee \mathbf{R}) \equiv (\mathbf{P} \wedge \mathbf{Q}) \vee (\mathbf{P} \wedge \mathbf{R})$

Validity and satisfiability

A sentence is **valid** if it is true in **all** models,
e.g., *True*, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the **Deduction Theorem**:
 $KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

A sentence is **satisfiable** if it is true in **some** model
e.g., $A \vee B$, C

A sentence is **unsatisfiable** if it is true in **no** models
e.g., $A \wedge \neg A$

Satisfiability is connected to inference via the following:
 $KB \models \alpha$ if and only if $(KB \wedge \neg \alpha)$ is unsatisfiable

Scheduling people to work in shifts at a hospital

- Some people don't work at night

- No one can work more than x hours a week

- Some pairs of people can't be on the same shift

Is there assignment of people to shifts that satisfy all constraints?

Inference rules in PL

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

And-elimination: from a conjunction any conjunction can be inferred:

$$\frac{\alpha \wedge \beta}{\alpha}$$

Example:

Statement-1: "If I am sleepy then I go to bed" $\Rightarrow P \rightarrow Q$


Statement-2: "I am sleepy" $\Rightarrow P$

Conclusion: "I go to bed." $\Rightarrow Q$.

Hence, we can say that, if $P \rightarrow Q$ is true and P is true then Q will be true.

Proof by Truth table:

P	Q	$P \rightarrow Q$
0	0	0
0	1	1
1	0	0
1	1	1



Modus Tollens:

The Modus Tollens rule state that if $P \rightarrow Q$ is true and $\neg Q$ is true, then $\neg P$ will also true.

$$\text{Notation for Modus Tollens: } \frac{P \rightarrow Q, \neg Q}{\neg P}$$

Statement-1: "If I am sleepy then I go to bed" $\Rightarrow P \rightarrow Q$

Statement-2: "I do not go to the bed." $\Rightarrow \neg Q$

Statement-3: Which infers that "**I am not sleepy**" $\Rightarrow \neg P$

Proof by Truth table:

P	Q	$\neg P$	$\neg Q$	$P \rightarrow Q$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	0
1	1	0	0	1

Hypothetical Syllogism:

The Hypothetical Syllogism rule states that if $P \rightarrow R$ is true whenever $P \rightarrow Q$ is true, and $Q \rightarrow R$ is true.

$$\frac{p \rightarrow q \quad q \rightarrow r}{\therefore p \rightarrow r}$$

Example:

Statement-1: If you have my home key then you can unlock my home.

$$P \rightarrow Q$$

Statement-2: If you can unlock my home then you can take my money.

$$Q \rightarrow R$$

Conclusion: If you have my home key then you can take my money.

$$P \rightarrow R$$

P	Q	R	$P \rightarrow Q$	$Q \rightarrow R$	$P \rightarrow R$
0	0	0	1	1	1
0	0	1	1	1	1
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	1	0	0
1	1	1	1	1	1

Disjunctive Syllogism:

The Disjunctive syllogism rule state that if $P \vee Q$ is true, and $\neg P$ is true, then Q will be true.

Notation of Disjunctive syllogism:
$$\frac{P \vee Q, \neg P}{Q}$$

Example:


Statement-1: Today is Sunday or Monday. $\implies P \vee Q$

Statement-2: Today is not Sunday. $\implies \neg P$

Conclusion: Today is Monday. $\implies Q$

Proof by truth-table:

P	Q	$\neg P$	$P \vee Q$
0	0	1	0
0	1	1	1
1	0	0	1
1	1	0	1




Simplification:

The simplification rule state that if **$P \wedge Q$** is true, then **Q or P** will also be true.

Notation of Simplification rule: $\frac{P \wedge Q}{Q}$ Or $\frac{P \wedge Q}{P}$

Proof by Truth-Table:

P	Q	$P \wedge Q$
0	0	0
1	0	0
0	1	0
1	1	1



Resolution:

The Resolution rule state that if $P \vee Q$ and $\neg P \wedge R$ is true, then $Q \vee R$ will also be true.

$$\text{Notation of Resolution} \frac{P \vee Q, \neg P \wedge R}{Q \vee R}$$

Proof by Truth-Table:

P	$\neg P$	Q	R	$P \vee Q$	$\neg P \wedge R$	$Q \vee R$
0	1	0	0	0	0	0
0	1	0	1	0	0	1
0	1	1	0	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	0	0
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	0	1	1	1	0	1

Predicate logic or First-order predicate logic.

- Objects: A, B, people, numbers, colors, wars, theories,
- Relations: It can be unary relation such as: red, round, is adjacent, or n-any relation such as: the sister of, brother of, has color, comes between
- Function: Father of, best friend, third inning of, end of,

Basic Elements of First-order logic:

Constant	1, 2, A, John, Mumbai, cat,....
Variables	x, y, z, a, b,....
Predicates	Brother, Father, >,....
Function	sqrt, LeftLegOf,
Connectives	\wedge , \vee , \neg , \Rightarrow , \Leftrightarrow
Equality	$=$
Quantifier	\forall , \exists

Atomic sentences:

- Atomic sentences are the most basic sentences of first-order logic. These sentences are formed from a predicate symbol followed by a parenthesis with a sequence of terms.
- We can represent atomic sentences as **Predicate (term1, term2,, term n)**.

Example: Ravi and Ajay are brothers: => Brothers(Ravi, Ajay).

Chinky is a cat: => cat (Chinky).

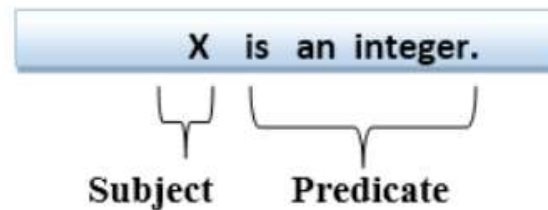
Complex Sentences:

- Complex sentences are made by combining atomic sentences using connectives.

First-order logic statements can be divided into two parts:

- Subject:** Subject is the main part of the statement.
- Predicate:** A predicate can be defined as a relation, which binds two atoms together in a statement.

Consider the statement: "x is an integer.", it consists of two parts, the first part x is the subject of the statement and second part "is an integer," is known as a predicate.



Quantifiers in First-order logic:

- A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.
- These are the symbols that permit to determine or identify the range and scope of the variable in the logical expression. There are two types of quantifier:
 - a. **Universal Quantifier, (for all, everyone, everything)**
 - b. **Existential quantifier, (for some, at least one).**

Universal Quantifier:

If x is a variable, then $\forall x$ is read as:

- **For all x**
- **For each x**
- **For every x .**

• In universal quantifier, $\forall x \forall y$ is similar to $\forall y \forall x$.

Example:

All man drink coffee.

$\forall x \text{ man}(x) \rightarrow \text{drink}(x, \text{coffee}).$

Existential Quantifier:

- Existential quantifiers are the type of quantifiers, which express that the statement within its scope is true for at least one instance of something.
- If x is a variable, then existential quantifier will be $\exists x$ or $\exists(x)$.
- In Existential quantifier we always use AND or Conjunction symbol (\wedge).

Example:

Some boys are intelligent.

- $\exists x: \text{boy}(x) \wedge \text{intelligent}(x)$
 - In Existential quantifier, $\exists x \exists y$ is similar to $\exists y \exists x$.
 - $\exists x \forall y$ is not similar to $\forall y \exists x$.

- Everyone likes some kind of food
- There is a kind of food that everyone likes
- Someone likes all kinds of food
- Every food has someone who likes it

- Everyone likes some kind of food
 - $\forall P \exists F \text{ likes}(P, F)$
- There is a kind of food that everyone likes
 - $\exists F \forall P \text{ likes}(P, F)$
- Someone likes all kinds of food
 - $\exists P \forall F \text{ likes}(P, F)$
- Every food has someone who likes it
 - $\forall F \exists P \text{ likes}(P, F)$

DEFINITION

FIRST-ORDER PREDICATE CALCULUS

First-order predicate calculus allows quantified variables to refer to objects in the domain of discourse and not to predicates or functions.

DEFINITION

SATISFY, MODEL, VALID, INCONSISTENT

For a predicate calculus expression X and an interpretation I :

If X has a value of T under I and a particular variable assignment, then I is said to *satisfy* X .

If I satisfies X for all variable assignments, then I is a *model* of X .

X is *satisfiable* if and only if there exist an interpretation and variable assignment that satisfy it; otherwise, it is *unsatisfiable*.

A set of expressions is *satisfiable* if and only if there exist an interpretation and variable assignment that satisfy every element.

If a set of expressions is not satisfiable, it is said to be *inconsistent*.

If X has a value T for all possible interpretations, X is said to be *valid*.

Example:

$\exists x p(x) \wedge \sim p(x)$ is inconsistent

$\forall x p(x) \vee \sim p(x)$ is valid

DEFINITION

PROOF PROCEDURE

A *proof procedure* is a combination of an inference rule and an algorithm for applying that rule to a set of logical expressions to generate new sentences.

FC: 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 Col. West, who is an American.
- Prove that Col. West is a criminal.

FC: EXAMPLE KNOWLEDGE BASE

- ...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

$\exists x$ Owns(Nono, x) \wedge Missiles(x)

Owns(Nono, M_1) and Missile(M_1)

- ...all of its missiles were sold to it by Col. West

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

- Missiles are weapons

Missile(x) \Rightarrow Weapon(x)

FC: EXAMPLE KNOWLEDGE BASE

- An enemy of America counts as “hostile”
Enemy(x, America) \Rightarrow Hostile(x)
- Col. West who is an American
American(Col. West)
- The country Nono, an enemy of America
Enemy(Nono, America)

FC: EXAMPLE KNOWLEDGE BASE

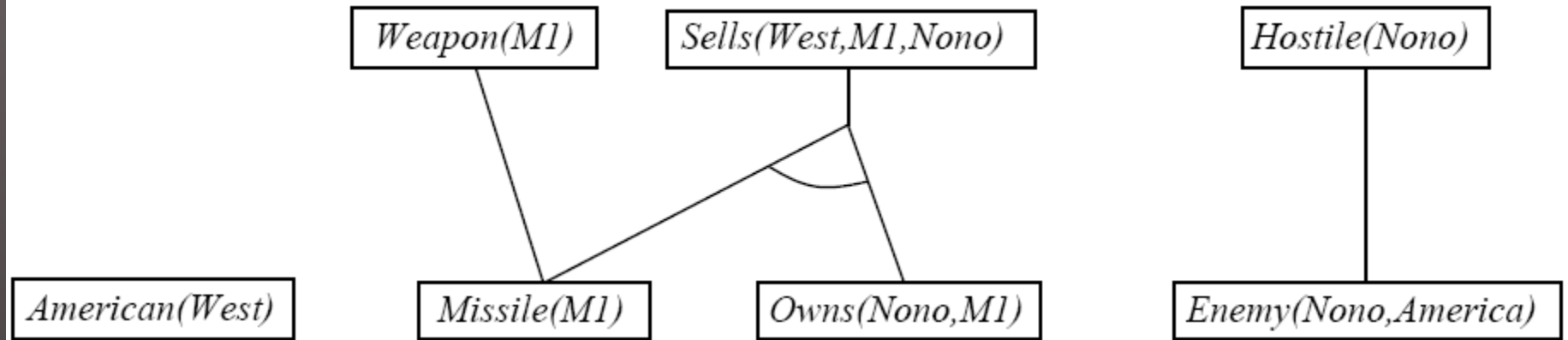
American(West)

Missile(M1)

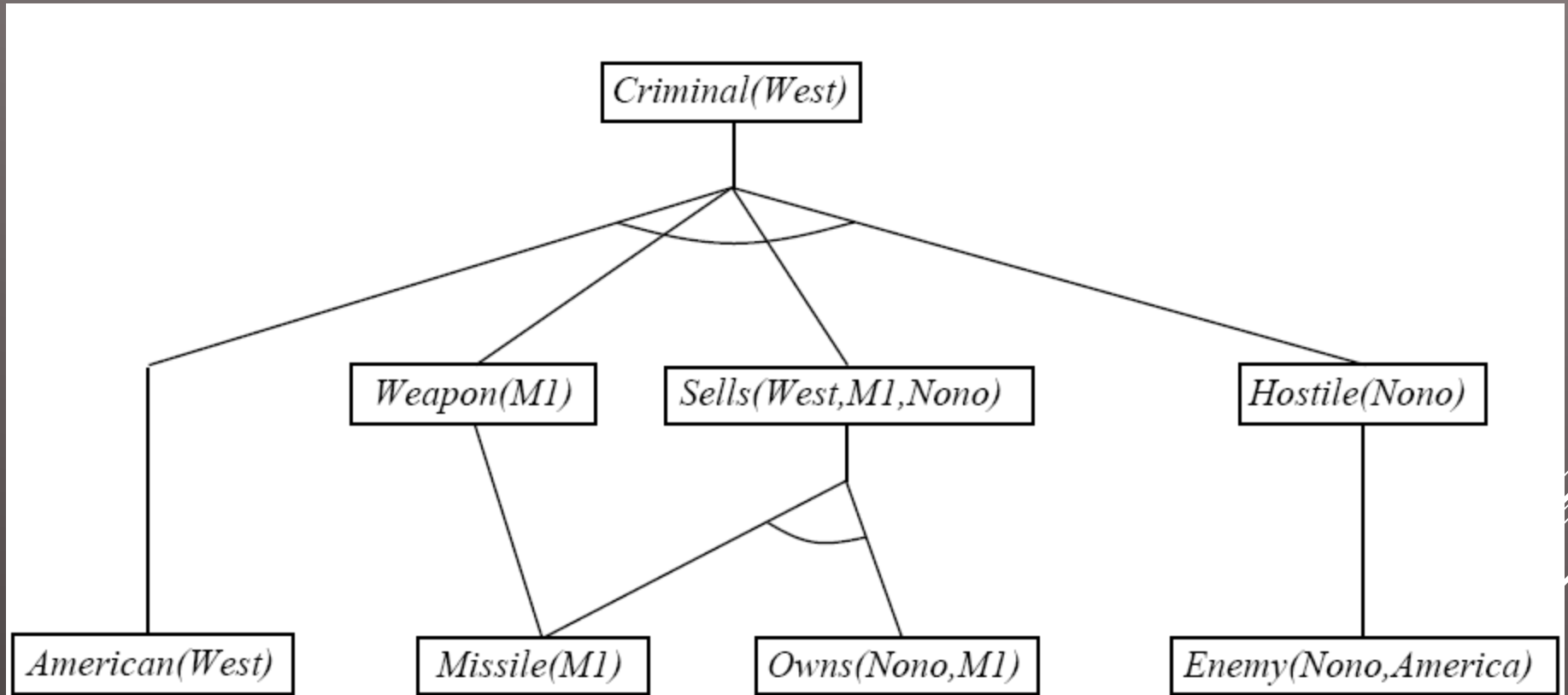
Owns(Nono,M1)

Enemy(Nono,America)

FC: EXAMPLE KNOWLEDGE BASE



FC: EXAMPLE KNOWLEDGE BASE



DEFINITION

LOGICALLY FOLLOWS, SOUND, and COMPLETE

A predicate calculus expression X *logically follows* from a set S of predicate calculus expressions if every interpretation and variable assignment that satisfies S also satisfies X .

An inference rule is *sound* if every predicate calculus expression produced by the rule from a set S of predicate calculus expressions also logically follows from S .

An inference rule is *complete* if, given a set S of predicate calculus expressions, the rule can infer every expression that logically follows from S .

DEFINITION

MODUS PONENS, MODUS TOLLENS, AND ELIMINATION, AND INTRODUCTION, and UNIVERSAL INSTANTIATION

If the sentences P and $P \rightarrow Q$ are known to be true, then *modus ponens* lets us infer Q .

Under the inference rule *modus tollens*, if $P \rightarrow Q$ is known to be true and Q is known to be false, we can infer $\neg P$.

And elimination allows us to infer the truth of either of the conjuncts from the truth of a conjunctive sentence. For instance, $P \wedge Q$ lets us conclude P and Q are true.

And introduction lets us infer the truth of a conjunction from the truth of its conjuncts. For instance, if P and Q are true, then $P \wedge Q$ is true.

Universal instantiation states that if any universally quantified variable in a true sentence is replaced by any appropriate term from the domain, the result is a true sentence. Thus, if a is from the domain of X , $\forall X p(X)$ lets us infer $p(a)$.

DEFINITION

MOST GENERAL UNIFIER (mgu)

If s is any unifier of expressions E , and g is the most general unifier of that set of expressions, then for s applied to E there exists another unifier s' such that $Es = Egs'$, where Es and Egs' are the composition of unifiers applied to the expression E .

INFERENCE WITH QUANTIFIERS

- Universal Instantiation:
 - Given $\forall x, \text{person}(x) \Rightarrow \text{likes}(x, \text{McDonalds})$
 - Infer $\text{person}(\text{John}) \Rightarrow \text{likes}(\text{John}, \text{McDonalds})$
- Existential Instantiation:
 - Given $\exists x, \text{likes}(x, \text{McDonalds})$
 - Infer $\Rightarrow \text{likes}(S1, \text{McDonalds})$
 - S1 is a “Skolem Constant” that is not found anywhere else in the KB and refers to (one of) the individuals that likes McDonalds.

UNIVERSAL INSTANTIATION

- Every instantiation of a universally quantified sentence is entailed by it:

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

- for any variable v and ground term g
 - ground term...a term with out variables

- Example:

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

- For any sentence α , variable v , and constant k that does not appear in the KB:

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

- Example:
 - $\exists x \text{ Crown}(x) \wedge \text{OnHead}(x, \text{John})$ yields:
 - $\text{Crown}(C_1) \wedge \text{OnHead}(C_1, \text{John})$
 - provided C_1 is a new constant (Skolem)

EXISTENTIAL INSTANTIATION

- UI can be applied several times to *add* new sentences
 - The KB is logically equivalent to the old
- EI can be applied once to *replace* the existential sentence
 - The new KB is not equivalent to the old but is satisfiable iff the old KB was satisfiable

REDUCTION TO PROPOSITIONAL INFERENCE

- Use instantiation rules to create relevant propositional facts in the KB, then use propositional reasoning
- Problems:
 - May generate infinite sentences when it cannot prove
 - Will generate many irrelevant sentences along the way!

REDUCTION TO PROPOSITIONAL INFERENCE

- Suppose the KB had the following sentence

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

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

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

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

REDUCTION TO PROPOSITIONAL INFERENCE

- Instantiating the universal sentence in all possible ways...
 - 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: propositional symbols are...
 - King(John), Greedy(John), Evil(John), King(Richard),
etc...

PROPOSITIONALIZATION

- Every FOL KB can be *propositionalized* so as to preserve entailment
 - A ground sentence is entailed by the new KB iff it is entailed by the original KB
- **Idea:** propositionalize KB and query, apply resolution, return result
- **Problem:** with function symbols, there are infinitely many ground terms
 - For example, $\text{Father}(X)$ yields $\text{Father}(\text{John})$, $\text{Father}(\text{Father}(\text{John}))$, $\text{Father}(\text{Father}(\text{Father}(\text{John})))$, etc.

PROBLEMS WITH PROPOSITIONALIZATION

- Propositionalization tends to generate lots of irrelevant sentences
- Example

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

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

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

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

- Obvious that $\text{Evil}(\text{John})$ is true, but the fact $\text{Greedy}(\text{Richard})$ is irrelevant

UNIFICATION

- **Unification**: The process of finding all legal substitutions that make logical expressions look identical
- This is a recursive algorithm

UNIFICATION

- We can get the inference immediately if we can find a substitution θ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(y)$
- $\theta = \{x/\text{John}, y/\text{John}\}$ works
- Unify algorithm takes in two sentences and returns a unifier if one exists
- $\text{Unify}(\alpha, \beta) = \theta$ if $\text{Subst}(\alpha, \theta) = \text{subst}(\beta, \theta)$

UNIFICATION

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

- Standardizing apart- renaming variables to avoid name clashes

$Unify(Knows(John, x), Knows(y, OJ)) = \{x/OJ, y/John\}$

BASE CASES FOR UNIFICATION

- If two expressions are identical, the result is (NIL)
(succeed with empty unifier set)
- If two expressions are different constants, the result is
NIL (fail)
- If one expression is a variable *and is not contained in
the other*, the result is ((x /other-exp))

GENERALIZED MODUS PONENS

- This is a general inference rule for FOL that does not require instantiation

- Given:

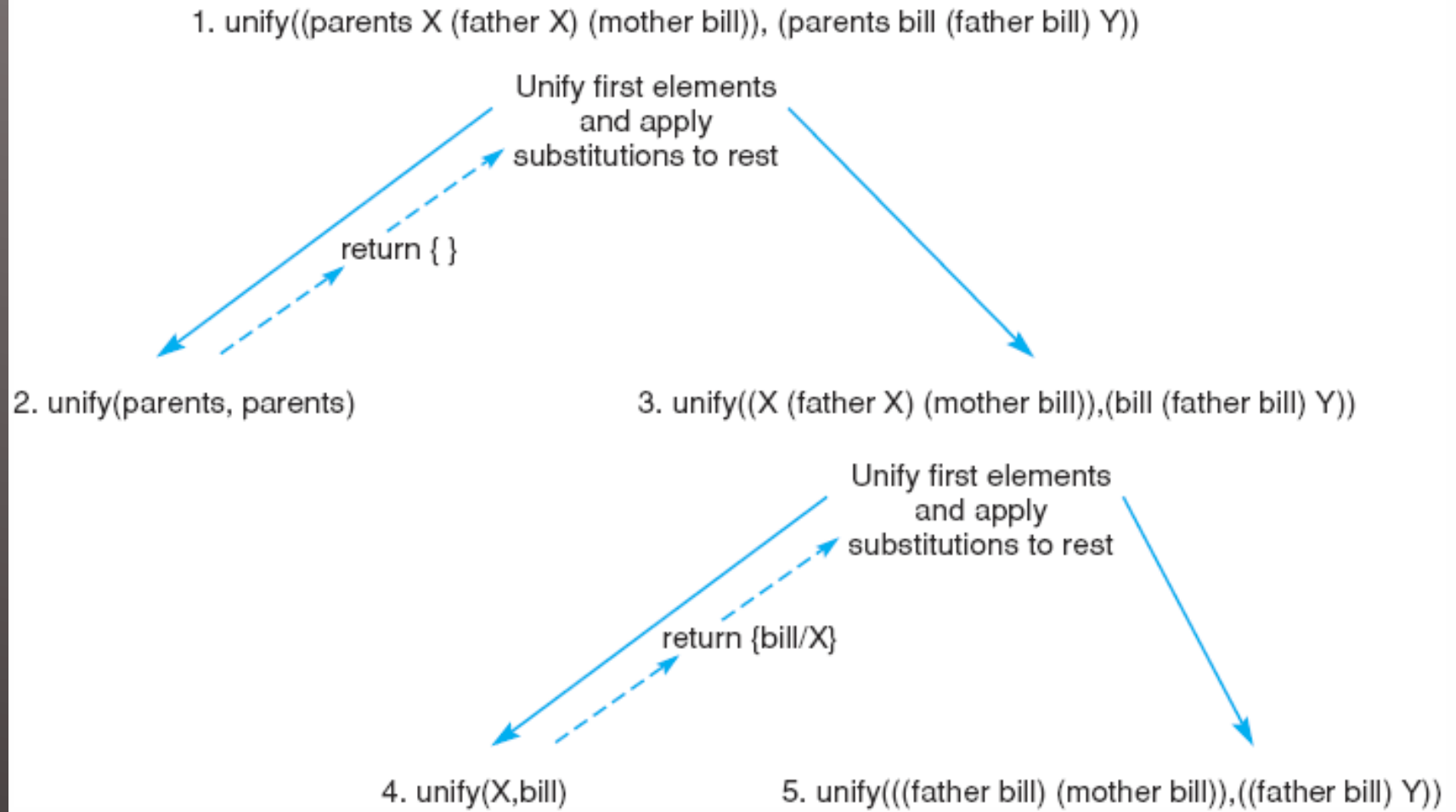
$$\frac{p_1', p_2', \dots, p_n', (p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q)}{\text{Subst}(q, \theta)}$$

$$p_1', p_2' \dots p_n', (p_1 \wedge \dots p_n) \Rightarrow q$$

$$\text{subst}(\theta, p_i') = \text{subst}(\theta, p_i) \text{ for all } p$$

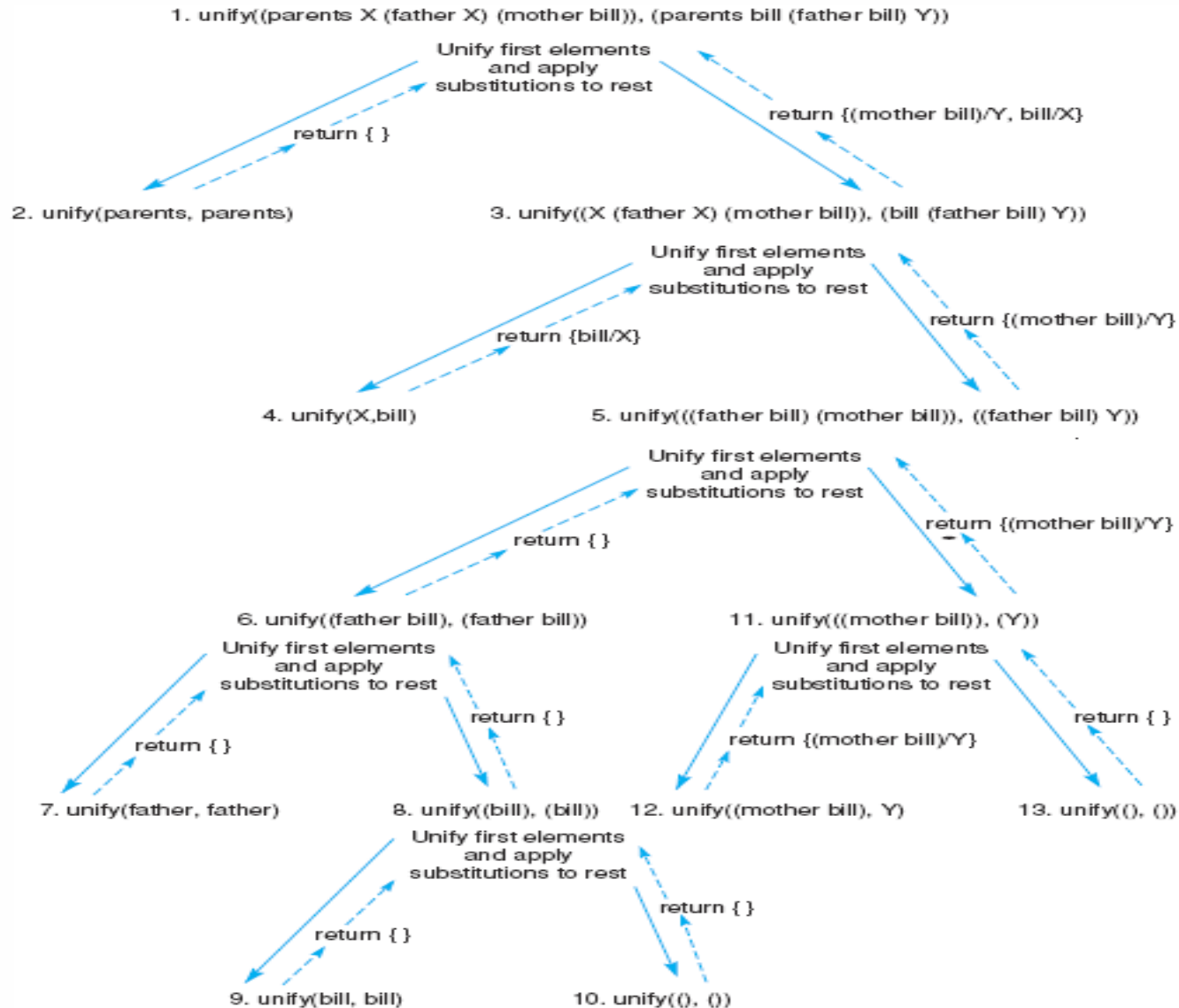
- Conclude:
 - $\text{Subst}(\theta, q)$

The unification of
(parents X (father X) (mother bill)) and (parents bill (father bill) Y).



The unification of

(parents X (father X) (mother bill)) and (parents bill (father bill) Y).



APPLICATION

A Logic-Based Financial Advisor

Logic-Based Application Example : Financial Advisor

As a final example, let's use FOL to represent and reason about a sample problem domain: financial analysis. Although simple, it illustrates many issues involved in real applications.

The function is to advise a user about whether to invest in stocks or savings. Some investors might want to split their money. Let's say that we have determined the following rules:

1. Individuals with inadequate savings should make increasing savings their top priority, regardless of income.
2. Individuals with adequate savings and adequate income should consider stocks, which are riskier but potentially more profit.
3. Individuals with a lower income and already have adequate savings should split their income between savings and stocks.

Our rule is to have at least \$5000 in the bank for each dependent. An adequate income must be steady and supply at least \$15000 per year plus \$4000 for each dependent.

1. **savings_account(inadequate) \rightarrow investment(savings).**
2. **savings_account(adequate) \wedge income(adequate) \rightarrow investment(stocks).**
3. **savings_account(adequate) \wedge income(inadequate) \rightarrow investment(combination).**
4. **\forall amount_saved(X) $\wedge \exists Y$ (dependents(Y) \wedge greater(X, minsavings(Y))) \rightarrow savings_account(adequate).**
5. **$\forall X$ amount_saved(X) $\wedge \exists Y$ (dependents(Y) \wedge \neg greater(X, minsavings(Y))) \rightarrow savings_account(inadequate).**
6. **$\forall X$ earnings(X, steady) $\wedge \exists Y$ (dependents(Y) \wedge greater(X, minincome(Y))) \rightarrow income(adequate).**
7. **$\forall X$ earnings(X, steady) $\wedge \exists Y$ (dependents(Y) \wedge \neg greater(X, minincome(Y))) \rightarrow income(inadequate).**
8. **$\forall X$ earnings(X, unsteady) \rightarrow income(inadequate).**
9. **amount_saved(22000).**
10. **earnings(25000, steady).**
11. **dependents(3).**