

# Predicate Calculus

## Lecture three

Assist. Prof. Dr. Asaad Sabah Hadi

# Knowledge Representation (KR)

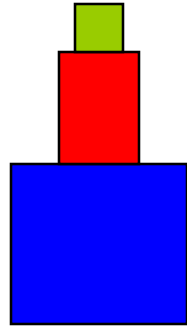
---

**Given the world**

- **Express the general facts or beliefs using a language**
- **Determine what else we should (not) believe**

# Example

---



## Given:

- “The red block is above the blue block”
- “The green block is above the red block”

## Infer:

- “The green block is above the blue block”
- “The blocks form a tower”

# A KR language needs to be

---

- expressive
- unambiguous
- flexible

# The inference procedures need to be

---

- **Correct (sound)**
- **Complete**
- **Efficient**

# Candidates (for now)

---

- English (natural language)
- Java (programming language)
- Logic (special KR language)

# Logic consists of

---

- A language  
which tells us how to build up sentences  
in the language (i.e., *syntax*), and  
and what the sentences mean  
(i.e., *semantics*)
- An inference procedure  
which tells us which sentences are valid  
inferences from other sentences

# Propositional logic

---

The symbols of propositional calculus are  
the propositional symbols:

$P, Q, R, S, \dots$

the truth symbols:

true, false

and connectives:

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



# Propositional Calculus Sentences

---

Every propositional symbol and truth symbol is a *sentence*.

Examples: true, P, Q, R.

The *negation* of a sentence is a sentence.

Examples:  $\neg P$ ,  $\neg$  false.

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

Example:  $P \wedge \neg P$

# Propositional Calculus Sentences

## (cont'd)

---

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

Example:  $P \vee \neg P$

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

Example:  $P \rightarrow Q$

The **equivalence** of two sentences is a sentence.

Example:  $P \vee Q \equiv R$

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

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

# Propositional calculus semantics (cont'd)

---

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 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 $P, Q, R$

---

$$\neg (\neg P) \equiv P$$

$$(P \vee Q) \equiv (\neg P \rightarrow Q)$$

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

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

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

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

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

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

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

# Truth table for the operator $\wedge$

---

<b>P</b>	<b>Q</b>	<b><math>P \wedge Q</math></b>
<b>T</b>	<b>T</b>	<b>T</b>
<b>T</b>	<b>F</b>	<b>F</b>
<b>T</b>	<b>T</b>	<b>F</b>
<b>T</b>	<b>F</b>	<b>F</b>

# Truth table demonstrating the equivalence of $\neg P \vee Q$ and $P \Rightarrow Q$

---

P	Q	$\neg P$	$\neg P \vee Q$	$P \Rightarrow Q$	$(\neg P \vee Q) = (P \Rightarrow Q)$
T	T	F	T	T	T
T	F	F	F	F	T
F	T	T	T	T	T
F	F	T	T	T	T

# Proofs in propositional calculus

---

**If it is sunny today, then the sun shines on the screen. If the sun shines on the screen, the blinds are brought down. The blinds are not down.**

**Is it sunny today?**

**P: It is sunny today.**

**Q: The sun shines on the screen.**

**R: The blinds are down.**

**Premises:  $P \rightarrow Q$ ,  $Q \rightarrow R$ ,  $\neg R$**

**Question: P**



# Prove using a truth table

---

Variables			Premises			Trial Conclusions	
P	Q	R	$P \rightarrow Q$	$Q \rightarrow R$	$\neg R$	P	$\neg P$
T	T	T	T	T	F	T	F
T	T	F	T	F	T	T	F
T	F	T	F	T	F	T	F
T	F	F	F	T	T	T	F
F	T	T	T	T	F	F	T
F	T	F	T	F	T	F	T
F	F	T	T	T	F	F	T
F	F	F	T	T	T	F	T

# Propositional calculus is cumbersome

---

**If it is sunny today, then the sun shines on the screen. If the sun shines on the screen, the blinds are brought down. The blinds are not down.**

**Is it sunny today?**

**- - -**

**If it is sunny on a particular day, then the sun shines on the screen. If the sun shines on the screen on a particular day, the blinds are brought down. The blinds are not down today.**

**Is it sunny today?**

# Represent in predicate calculus

---

If it is sunny on a particular day, then the sun shines on the screen [on that day]. If the sun shines on the screen on a particular day, the blinds are down [on that day]. The blinds are not down today.

Is it sunny today?

Premises:

$\forall D \text{ sunny}(D) \rightarrow \text{screen-shines}(D)$

$\forall D \text{ screen-shines}(D) \rightarrow \text{blinds-down}(D)$

$\neg \text{blinds-down}(\text{today})$

Question:  $\text{sunny}(\text{today})$

# Can also use functions

---

**A person's mother is that person's parent.**

**$\forall X \text{ person}(X) \rightarrow \text{parent}(\text{mother-of}(X), X)$**

**There are people who think this class is cool.**

**$\exists X \text{ person}(X) \wedge T(X)$**

**Some computers have mice connected on the USB.**

**$\exists X \text{ computer}(X) \wedge \text{USB\_conn}(X, \text{mouse\_of}(X))$**

# Predicate calculus symbols

---

The set of letters (both uppercase and lowercase): A ... Z, a ... z.

The set of digits: 0 ... 9

The underscore: \_

Needs to start with a letter.

# Symbols and terms

---

1. ***Truth symbols*** true and false (these are reserved symbols)
2. ***Constant symbols*** are symbol expressions having the first character lowercase.

E.g., today, fisher

3. ***Variable symbols*** are symbol expressions beginning with an uppercase character.

E.g., X, Y, Z, Building

4. ***Function symbols*** are symbol expressions having the first character lowercase.

Arity: number of elements in the domain

E.g., mother-of (bill); maximum-of (7,8)

# Symbols and terms (cont'd)

---

A **function expression** consists of a function constant of arity  $n$ , followed by  $n$  terms,  $t_1, t_2, \dots, t_n$ , enclosed in parentheses and separated by commas.

E.g., mother-of(mother-of(joe))

maximum(maximum(7, 18), add\_one(18))

A **term** is either a constant, variable, or function expression.

E.g. color\_of(house\_of(neighbor(joe)))

house\_of(X)

# Predicates and atomic sentences

---

***Predicate symbols*** are symbols beginning with a lowercase letter. Predicates are special functions with true/false as the range.

**Arity:** number of arguments

An ***atomic sentence*** is a predicate constant of arity  $n$ , followed by  $n$  terms,  $t_1, t_2, \dots, t_n$ , enclosed in parentheses and separated by commas.

The truth values, true and false, are also atomic sentences.



# Examples

---

**greater\_than(2, 3)**

Predicate  
symbol

term (constant)

**mother\_of(joe, susan)**

**mother\_of(sister\_of(joe), susan)**

# Predicate calculus sentences

---

**Every atomic sentence is a sentence.**

**1. If  $s$  is a sentence, then so is its negation,  $\neg s$ .**

**If  $s_1$  and  $s_2$  are sentences, then so is their**

**2. Conjunction,  $s_1 \wedge s_2$  .**

**3. Disjunction,  $s_1 \vee s_2$  .**

**4. Implication,  $s_1 \rightarrow s_2$  .**

**5. Equivalence,  $s_1 \equiv s_2$  .**

# Predicate calculus sentences (cont'd)

---

If  $X$  is a variable and  $s$  is a sentence, then so are

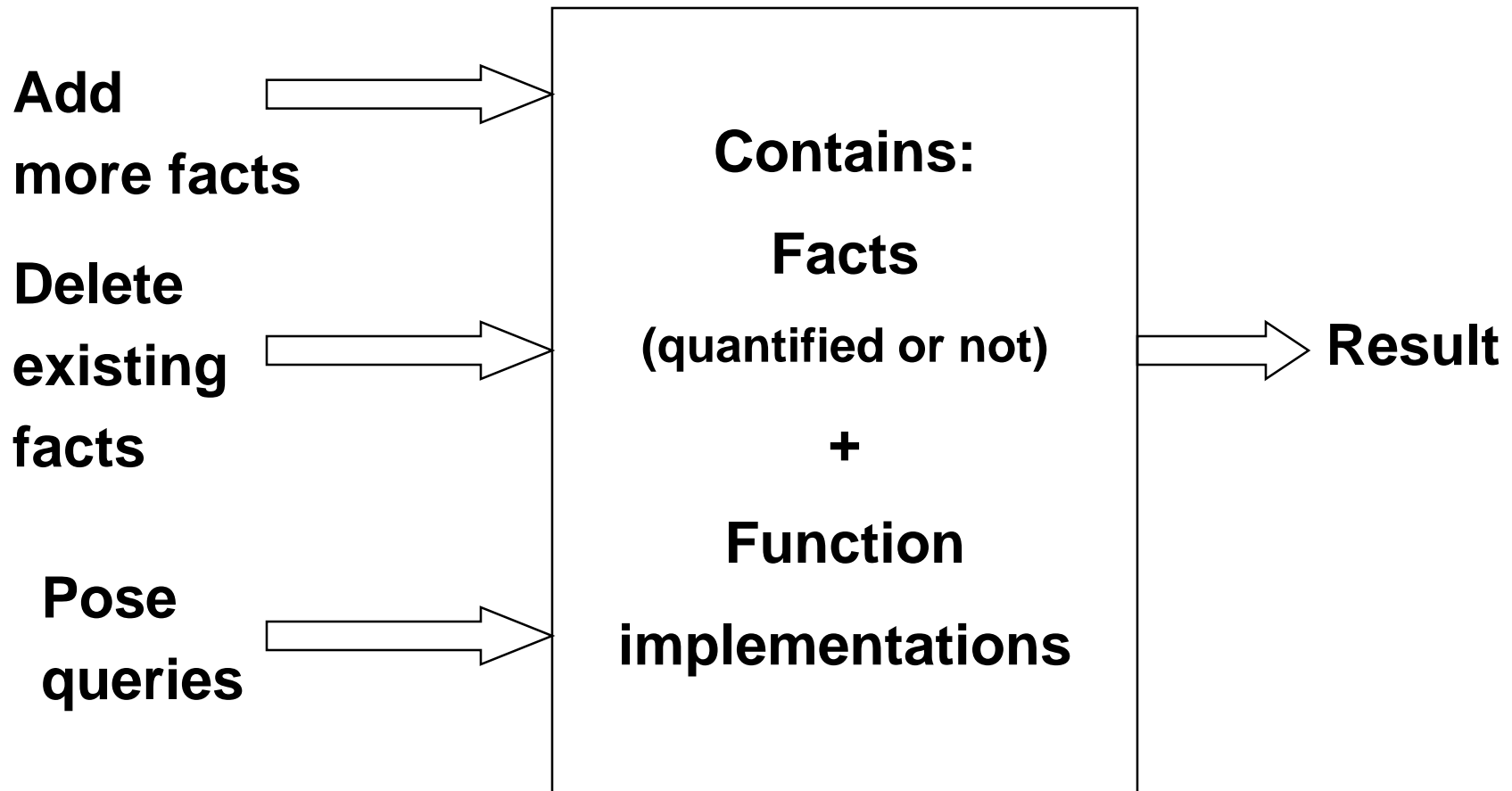
6.  $\forall X s$ .

7.  $\exists X s$ .

Remember that logic sentences evaluate to true or false, therefore only such objects are atomic sentences. Functions are not atomic sentences.

# A logic-based Knowledge Base (KB)

---



# Interpretation

---

Let the domain  $D$  be a nonempty set.

An *interpretation* over  $D$  is an assignment of the entities of  $D$  to each of the constant, variable, predicate, and function symbols of a predicate calculus expression:

1. Each constant is assigned an element of  $D$ .
2. Each variable is assigned to a nonempty subset of  $D$  (*allowable substitutions*).
3. Each function  $f$  of arity  $m$  is defined ( $D^m$  to  $D$ ).
4. Each predicate of arity  $n$  is defined ( $D^n$  to  $\{T, F\}$ ).

# **How to compute the truth value of predicate calculus expressions**

**Assume an expression  $E$  and an interpretation  $I$  over  $E$  over a nonempty domain  $D$ . The truth value for  $E$  is determined by:**

- 1. The value of a constant is the element of  $D$  it is assigned to by  $I$ .**
- 2. The value of a variable is the set of elements of  $D$  it is assigned to by  $I$ .**
- 3. The value of a function expression is that element of  $D$  obtained by evaluating the function for the parameter values assigned by the interpretation.**

# **How to compute the truth value of predicate calculus expressions (cont'd)**

**4. The value of the truth symbol “true” is T, and “false” is F.**

**5. The value of an atomic sentence is either T or F, as determined by the interpretation I.**

**6. The value of the negation of a sentence is T if the value of the sentence is F, and F if the value of the sentence is T.**

**7. The value of the conjunction of two sentences is T, if the value of both sentences is T and F otherwise.**

**8-10. The truth value of expressions using  $\vee$ ,  $\rightarrow$ , and  $\equiv$  is determined as defined in Section 2.1.2.**

# **How to compute the truth value of predicate calculus expressions (cont'd)**

**Finally, for a variable  $X$  and a sentence  $S$  containing  $X$ :**

**11. The value of  $\forall X S$  is T if  $S$  is T for all assignments to  $X$  under  $I$ , and it is F otherwise.**

**12. The value of  $\exists X S$  is T if there is an assignment to  $X$  under  $I$  such that  $S$  is T, and it is F otherwise**



# Revisit $\forall$ and $\exists$

---

**A person's mother is that person's parent.**

**$\forall X \text{ person}(X) \rightarrow \text{parent}(\text{mother-of}(X), X)$**

**vs.**

**$\forall X \text{ person}(X) \wedge \text{parent}(\text{mother-of}(X), X)$**

**I: joe, jane are people**

**fido is a dog**

**person(joe) is T,**

**person(jane) is T**

**person(fido) is F,**

**dog(fido) is T**

**mother-of(joe) is jane**

# Revisit $\forall$ and $\exists$ (cont'd)

---

There are people who think this class is cool.

$$\exists X \text{ person } (X) \wedge T (X)$$

vs.

$$\exists X \text{ person } (X) \rightarrow T (X)$$

I: joe, jane are people

fido is a dog

person (joe) is T,

person (jane) is T

person (fido) is F,

dog (fido) is T

mother-of (joe) is jane

# 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.

John likes to eat everything.

$\forall X \text{ food}(X) \rightarrow \text{likes}(\text{john}, X)$

John likes at least one dish Jane likes.

$\exists F \text{ food}(F) \wedge \text{likes}(\text{jane}, F) \wedge \text{likes}(\text{john}, F)$

John “does” everything Jane does.

$\forall P P(\text{Jane}) \rightarrow P(\text{john})$     This is not first-order.

# Order of quantifiers matters

---

**Everybody likes some food.**

**There is a food that everyone likes.**

**Whenever someone likes at least one spicy dish, they're happy.**

# Order of quantifiers matters

---

**Everybody likes some food.**

$$\forall X \exists F \text{ food}(F) \wedge \text{likes}(X, F)$$

**There is a food that everyone likes.**

$$\exists F \forall X \text{ food}(F) \wedge \text{likes}(X, F)$$

**Whenever someone eats a spicy dish, they're happy.**

$$\forall X \exists F \text{ food}(F) \wedge \text{spicy}(F) \wedge \text{eats}(X, F) \rightarrow \text{happy}(X)$$

# Examples

---

**John's meals are spicy.**

**Every city has a dogcatcher who has been bitten by every dog in town.**

**For every set  $x$ , there is a set  $y$ , such that the cardinality of  $y$  is greater than the cardinality of  $x$ .**

# Examples

---

**John's meals are spicy.**

**$\forall X \text{ meal-of}(\text{John}, X) \rightarrow \text{spicy}(X)$**

**Every city has a dogcatcher who has been bitten by every dog in town.**

**$\forall T \exists C \forall D \text{ city}(C) \rightarrow ( \text{dogcatcher}(C, T) \wedge$   
 $(\text{dog}(D) \wedge \text{lives-in}(D, T) \rightarrow \text{bit}(D, C)) )$**

## Examples (cont'd)

---

**For every set  $x$ , there is a set  $y$ , such that the cardinality of  $y$  is greater than the cardinality of  $x$ .**

**$\forall X \exists Y \exists U \exists V \text{ set}(X) \rightarrow (\text{set}(Y) \wedge \text{cardinality}(X, U) \wedge \text{cardinality}(Y, V) \wedge \text{greater-than}(V, U))$**



# The role of the knowledge engineer

---

**fisher-hall-is-a-building**

**ee-is-a-building**

**building (fisher)**

**building (ee)**

**white-house-on-the-corner-is-a-building**

**green (fisher)**

**color (fisher, green)**

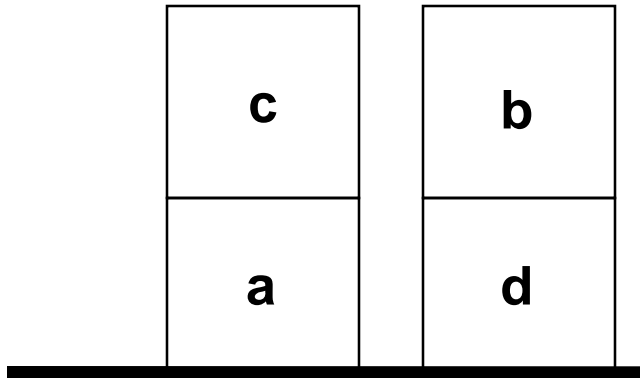
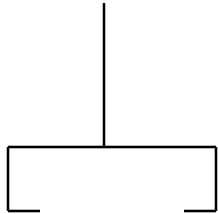
**holds (color, fisher, green)**

**holds (color, fisher, green, jan-2003)**

**holds (color, fisher, blue, jul-2003)**

# Blocks world

---



**on (c,a)**

**on(b,d)**

**ontable(a)**

**ontable(d)**

**clear(b)**

**clear(c)**

**hand\_empty**

# Blocks world example

---

All blocks on top of blocks that have been moved or that are attached to blocks that have been moved have also been moved.

$$\forall X \forall Y (\text{block}(X) \wedge \text{block}(Y) \wedge$$
$$(\text{on}(X,Y) \vee \text{attached}(X,Y)) \wedge \text{moved}(Y)) \rightarrow$$
$$\text{moved}(X)$$

# 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* iff there is an interpretation and variable assignment that satisfy it; otherwise it is *unsatisfiable*.

# Satisfy, model, valid, inconsistent (cont'd)

---

A set of expressions is *satisfiable* iff there is 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*.

# 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.

# 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  $S$ .

# Modus ponens and modus tollens

---

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

If the sentence  $P \rightarrow Q$  is known to be true, and the sentence  $Q$  is known to be false, *modus tollens* lets us infer  $\neg P$ .



# And elimination / and introduction

---

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

***And introduction*** allows us to 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

---

***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  $P(a)$ .

# Revisit the “sunny day” example

---

$\forall D \text{ sunny}(D) \rightarrow \text{screen-shines}(D)$

$\forall D \text{ screen-shines}(D) \rightarrow \text{blinds-down}(D)$

$\neg \text{blinds-down}(\text{today})$

Question:  $\text{sunny}(\text{today})$

Use “unification” and modus tollens:

$\text{sunny}(\text{today}) \rightarrow \text{screen-shines}(\text{today})$

$\text{screen-shines}(\text{today}) \rightarrow \text{blinds-down}(\text{today})$

$\neg \text{blinds-down}(\text{today})$

# Unification

---

**Make sentences look alike.**

**Unify  $p(a,X)$  and  $p(a,b)$**

**Unify  $p(a,X)$  and  $p(Y,b)$**

**Unify  $p(a,X)$  and  $p(Y, f(Y))$**

**Unify  $p(a,X)$  and  $p(X,b)$**

**Unify  $p(a,X)$  and  $p(Y,b)$**

**Unify  $p(a,b)$  and  $p(X, X)$**

# Unification examples

---

Unify  $p(a,X)$  and  $p(a,b)$

answer:  $b/X$

$p(a,b)$

Unify  $p(a,X)$  and  $p(Y,b)$

answer:  $a/Y, b/X$

$p(a,b)$

Unify  $p(a,X)$  and  $p(Y, f(Y))$

answer:  $a/Y, f(a)/X$

$p(a,f(a))$

# Unification examples (cont'd)

---

Unify  $p(a,X)$  and  $p(X,b)$

failure

Unify  $p(a,X)$  and  $p(Y,b)$

answer:  $a/Y, b/X$

$p(a,b)$

Unify  $p(a,b)$  and  $p(X,X)$

failure

Unify  $p(X, f(Y), b)$  and  $P(X, f(b), b)$

answer:  $b/Y$     this is an mgu(Most General Unifier)

$b/X, b/Y$     this is not an mgu

# 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  $E$ .**

**Basic idea: Commit to a substitution only if you have to; keep it as general as possible.**

# Unification algorithm

---

**Basic idea: can replace variables by:**

- **other variables**
- **constants**
- **function expressions**

**High level algorithm:**

- **Represent the expression as a list**
- **Process the list one by one**
  - **Determine a substitution (if necessary)**
  - **Apply to the rest of the list before proceeding**



# Examples with the algorithm

---

Unify  $p(a,X)$  and  $p(a,b)$

$(p\ a\ X)\ (p\ a\ b)$

Unify  $p(a,X)$  and  $p(Y, f(Y))$

$(p\ a\ X)\ (p\ Y\ (f\ Y))$

Unify  $parents(X, father(X), mother(bill))$  and  
 $parents(bill, father(bill), Y)$

$(parents\ X\ (father\ X)\ (mother\ bill))$

$(parents\ bill\ (father\ bill)\ Y)$

## function unify code

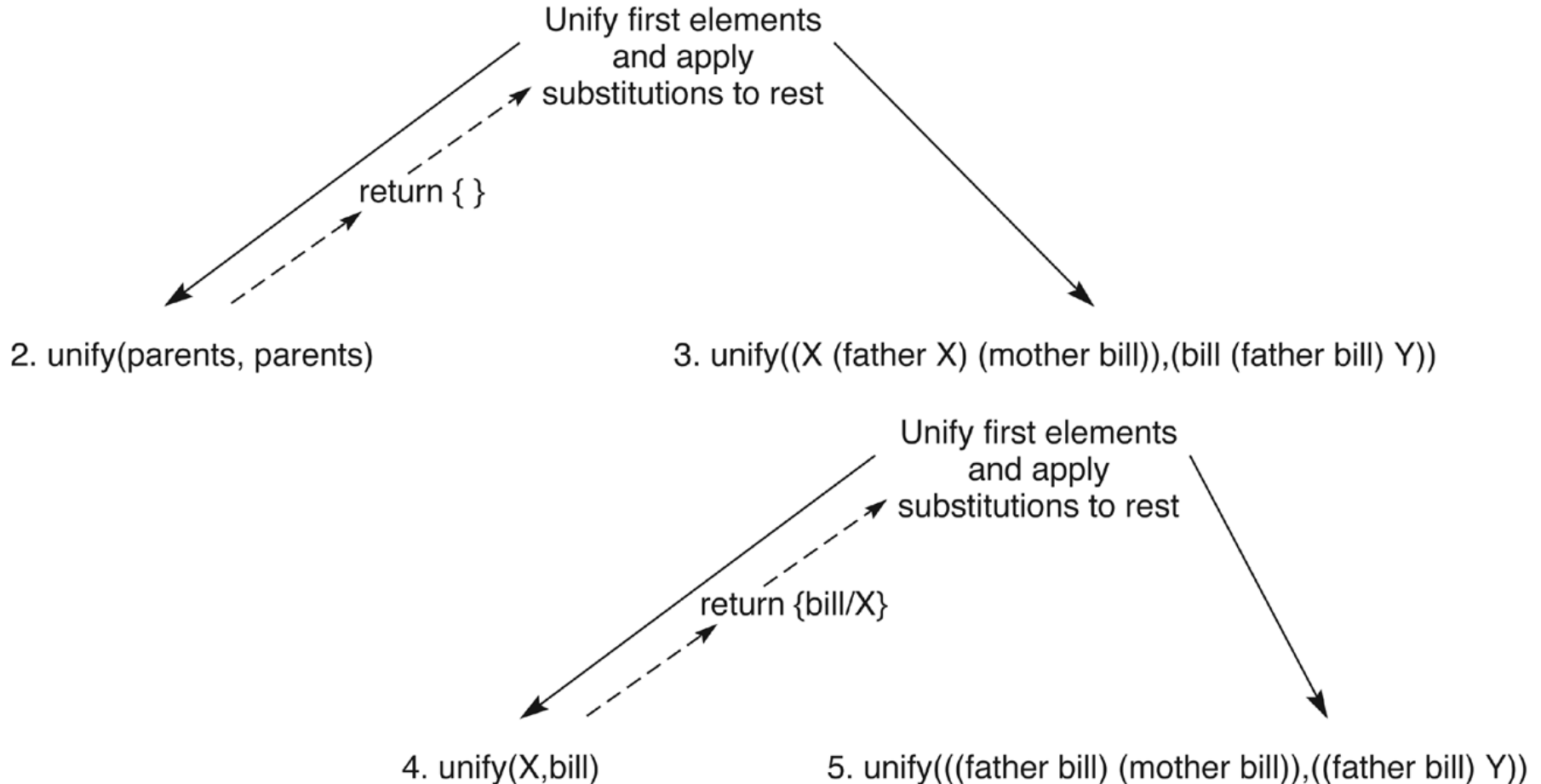
```
function unify(E1, E2);
begin
  case
    both E1 and E2 are constants or the empty list:           %recursion stops
      if E1 = E2 then return {}
      else return FAIL;
    E1 is a variable:
      if E1 occurs in E2 then return FAIL
      else return {E2/E1};
    E2 is a variable:
      if E2 occurs in E1 then return FAIL
      else return {E1/E2}
    either E1 or E2 are empty then return FAIL                 %the lists are of different sizes
    otherwise:                                                  %both E1 and E2 are lists
      begin
        HE1 := first element of E1;
        HE2 := first element of E2;
        SUBS1 := unify(HE1,HE2);
        if SUBS1 = FAIL then return FAIL;
        TE1 := apply(SUBS1, rest of E1);
        TE2 := apply (SUBS1, rest of E2);
        SUBS2 := unify(TE1, TE2);
        if SUBS2 = FAIL then return FAIL;
        else return composition(SUBS1,SUBS2)
      end
  end
end
end
```

%end case

# The book's example

---

1. unify((parents X (father X) (mother bill)), (parents bill (father bill) Y))



# Processed example

---

(parents X (father X) (mother bill)), (parents bill (father bill) Y)

parents =? parents    yes

return nil

(X (father X) (mother bill)), (bill (father bill) Y)

X =? bill    no, substitute

return {bill/X}

(bill (father bill) (mother bill)), (bill (father bill) Y)

bill =? bill    yes

return nil

# Processed example (cont'd)

---

( (father bill) (mother bill)), ( (father bill) Y)

(father bill), (father bill)

father =? father    yes

return nil

(bill) (bill)

bill =? bill    yes

return nil

# Processed example (cont'd)

---

(mother bill), Y

(mother bill) =? Y    no, substitute

return {(mother bill) / Y}

The set of unifying substitutions for

(parents X (father X) (mother bill)), (parents bill (father bill) Y)

is

{bill / X, (mother bill) / Y}.

The result is

(parents bill (father bill) (mother bill))

# A Logic-Based Financial Advisor

---

**Gives investment advice (savings account, or the stock market, or both).**

**Example “rule”:**

**If the amount in the savings account is inadequate, increasing this amount should be the first priority.**

# Sentences

---

1.  $\text{savings\_account}(\text{inadequate}) \rightarrow \text{investment}(\text{savings})$
2.  $\text{savings\_account}(\text{adequate}) \wedge \text{income}(\text{adequate}) \rightarrow \text{investment}(\text{stocks})$
3.  $\text{savings\_account}(\text{adequate}) \wedge \text{income}(\text{inadequate}) \rightarrow \text{investment}(\text{combination})$
4.  $\forall X \text{ amount\_saved}(X) \wedge \exists Y (\text{dependents}(Y) \wedge \text{greater}(X, \text{minsavings}(Y))) \rightarrow \text{savings\_account}(\text{adequate})$

Y is the number of dependents, minsavings is the number of dependents multiplied by 5000.



# Sentences (cont'd)

---

5.  $\forall X \text{ amount\_saved}(X) \wedge \exists Y (\text{dependents}(Y) \wedge \neg \text{greater}(X, \text{minsavings}(Y))) \rightarrow \text{savings\_account}(\text{inadequate})$

6.  $\forall X \text{ earnings}(X, \text{steady}) \wedge \exists Y (\text{dependents}(Y) \wedge \text{greater}(X, \text{minincome}(Y))) \rightarrow \text{income}(\text{adequate})$

7.  $\forall X \text{ earnings}(X, \text{steady}) \wedge \exists Y (\text{dependents}(Y) \wedge \neg \text{greater}(X, \text{minincome}(Y))) \rightarrow \text{income}(\text{inadequate})$

Minimum income is

$15,000 + (4000 * \text{number of dependents})$

8.  $\forall X \text{ earnings}(X, \text{unsteady}) \rightarrow \text{income}(\text{inadequate})$

# Sentences (cont'd)

---

9. amount\_saved(22000)

10. earnings(25000, steady)

11. dependents (3)

The knowledge base is an implicit  $\wedge$  of the sentences above.

Using 10, 11, and 7 we can infer

12. income(inadequate)

Using 9, 11, and 4, we can infer

13. savings\_account(adequate)

Using 12, 13, and 3, we can infer

14. investment(combination)