1

Terms

<term> ::= <constant> | <variable> | <functor>
<constant> ::= <atom starting w/lower case>
<variable> ::= <atom starting w/upper case>
<functor> ::= <constant> '(' <tlist> ')'
<tlist> ::= <term> { ',' <term> }

• Examples

```
a1 b fred                    constants
X  Yc3   Fred   _            variables
married(fred)
     married(Fred)           functors
```

• Ground terms -- terms with no variables

Terms are used in Prolog to represent symbolic structures. For example, lists are a simple kind of term, and using Prolog notation are represented with square brackets instead of round brackets. As well, list elements are separated by commas. For example:

Lisp: `(a b c)`
Prolog; `[a, b, c]`
but also `cons(a,cons(b,cons(c,nil)))`

'term' and 'structure' will be used synonymously in the sequel

©1993 John Mylopoulos

2

Substitution

• A substitution is a set of pairs $\{v_1/t_1, v_2/t_2, ..., v_n/t_n\}$ where v_i are distinct variables and t_i are terms not using v_j

Examples

{X/a} "substitute X with a"
{X/a, Y/b, Z/f(a, W)}

• Applying a substitution

Given a term t and a substitution σ, tσ is the term resulting from the application of the substitution σ to the term t

Examples
Let t = f(a, h(Y, b), X)

t{X/b} = f(a, h(Y, b), b)
t{X/b Y/f(Z)} = f(a, h(f(Z), b), b)
t{X/Z Y/f(Z, a)} = f(a, h(f(Z, a), b), Z)

σ need not include all variables in t and may include variables not in t

©1993 John Mylopoulos

3

Composition of Substitutions

$$t(\sigma \bullet \theta) = (t\theta)\sigma$$

For example,

$$f(X)\ [\ \{Z/a\} \bullet \{X/Z\}] = f(Z)\{Z/a\} = f(a)$$

Unification

Terms t_1 and t_2 are unified by σ iff $t_1\sigma = t_2\sigma$. Then σ is called a unifier of t_1, t_2

Examples

t_1	t_2	Unifier
f(b, c)	f(b, c)	{}
f(X, b)	f(a, Y)	{X/a, Y/b}
f(a, b)	f(c, d)	*
f(b, a)	f(X, X)	*
f(X, a)	f(Y, Y)	_____
f(X, g(X))	f(Y, Y)	won't unify

(see also occurscheck)

Before unifying, make sure the two terms have no common variables; if not, rename some variables

©1993 John Mylopoulos

4

What Unifies ?

More Examples:

			Unifier
[a,[b,c],d]	with	[a,X,Y]	{X/[b,c],Y/d}
[a,[b,U],d]	with	[a,X,Y]	{X/[b,U],Y/d }
[a,[b,X],d]	with	[a,V,Y]	{V/[b,X],Y/d}
[U,[b,U],a]	with	[a,X,V]	{U/a X/[b,a],V/a}
[a,b,b]	with	[a,X,X]	{X/b}
[a,b,Y]	with	[a,X,c]	{X/b,Y/c}
[a,b,Y]	with	[a,X,Z]	{X/b,Y/Z}

©1993 John Mylopoulos

5

Most General Unifier

In general, when a pair of terms unifies, it will have many unifiers. Among them, there is always at least one most general unifier (mgu) σ such that if σ' is another unifier of the same terms, then there is another substitution τ such that $\tau \bullet \sigma = \sigma'$

For example consider unifiers for:

[a, X, c] with [a, Y, c]

Here are some:

{X/b, Y/b}
{X/k, Y/k}
{X/Y}

The last one is the most general:

{Y/b} • {X/Y} = {X/b,Y/b}
{Y/k} • {X/Y) = {X/k,Y/k}

If two terms can be unified, then they have a unique mgu up to renaming of variables

©1993 John Mylopoulos

6

Finding the Most General Unifier

```
Recursive Procedure MGU(x, y)
    Begin x = y  ⇒ Return()
        Variable(x) ⇒ Return(MGUVar(x, y))
        Variable(y) ⇒ Return(MGUVar(y, x))
        Constant(x) ∨ Constant(y) ⇒ Return(False)
        ¬(FunctionSymbol(x) = FunctionSymbol(y))
                                  ⇒ Return(False)
        ¬(Length(x) = Length(y))  ⇒ Return(False)
        Begin  i ← 0
               g ← []
        Tag    i = Length(x) ⇒ Return(g)
               s ← MGU(part(x, i), Part(y, i))
               g ← g ∘ s
               x ← Substitute(x, g)
               y ←  Substitute(y, g)
               i ←  i + 1
               Goto Tag
        End
    End

Procedure MGUVar(v, e)
    Begin Includes(v, e) ⇒ Return(False)
          Return( {v/e})
    End
```


©1993 John Mylopoulos

7

Finding the Most General Unifier: Examples

f(X, g(a,Y), X) Z

f(X, g(a,Y), X) f(a, Z, b)

f(X, g(a,Y), X) f(Z, W, g(Z, Z))

©1993 John Mylopoulos

8

Prolog Syntax

- BNF

  <Clause> ::= <Literal> . | <HornClause> .
  <HornClause> ::= <Literal> :- <LList>
  <LIst> ::= <Literal> { , <Literal>}
  <Literal> ::= <Term>

- Examples

  father(john, sue).
  father(wodun, X).
  parent(X, Y) :- father(X, Y).
  gparent(X, Z) :- parent(X, Y), parent(Y, Z).

- How to read as Predicate Calculus

  father(john, sue)
  $(\forall X)$father(wodun, X)
  $(\forall X, Y)[\text{father}(X, Y) \Rightarrow \text{parent}(X, Y)]$
  $(\forall X, Y, Z)[\text{parent}(X, Y) \wedge \text{parent}(Y, Z) \Rightarrow \text{gparent}(X, Z)]$

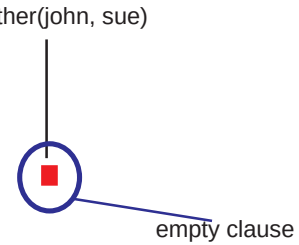
©1993 John Mylopoulos

9

Example: Consider Prog1

```
father(john, sue).                          (1)
father(wodun, X).                           (2)
parent(X, Y) :- father(X, Y).               (3)
gparent(X, Z) :- parent(X, Y), parent(Y, Z). (4)
```

• Goal 1: father(john, sue) (?- father(john, sue))



Hence the goal has been satisfied

• Looking for solutions to "achieve" or "satisfy" goals amounts to execution in Prolog programs.

• Proof trees amount to Prolog computations.

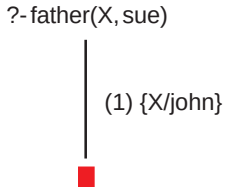
• Unification is the basic operation through which proof trees are generated

©1993 John Mylopoulos

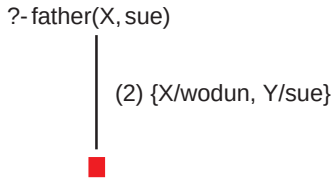
10


```
father(john, sue).                          (1)
father(wodun, X).                           (2)
parent(X, Y) :- father(X, Y).               (3)
gparent(X, Z) :- parent(X, Y), parent(Y, Z). (4)
```

• Goal 2: ?- father(X, sue)



More solutions?



©1993 John Mylopoulos

11

father(john, sue). (1)
father(wodun, X). (2)
parent(X, Y) :- father(X, Y). (3)
gparent(X, Z) :- parent(X, Y), parent(Y, Z). (4)

More examples:

?- father(bill, sue) ×

?- father(wodun, bill)

(2) {X/bill}

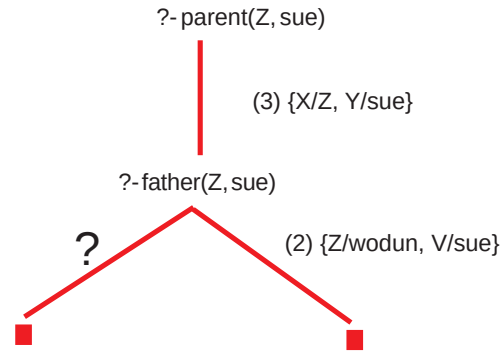
?- parent(john, sue)

(3) {X/john, Y/sue}

?- father(john, sue)

©1993 John Mylopoulos

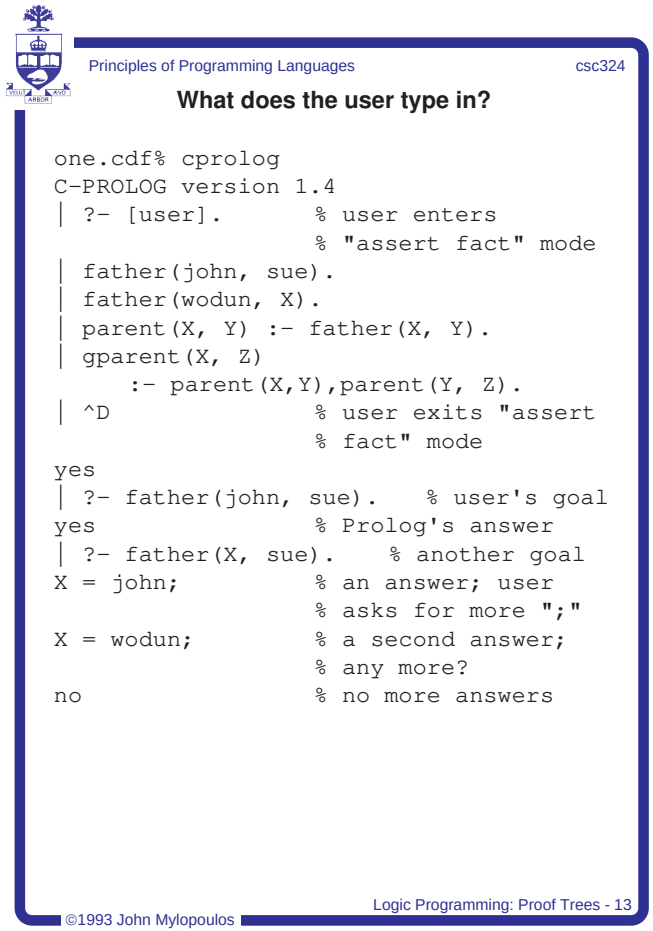
12



Proof Tree

©1993 John Mylopoulos

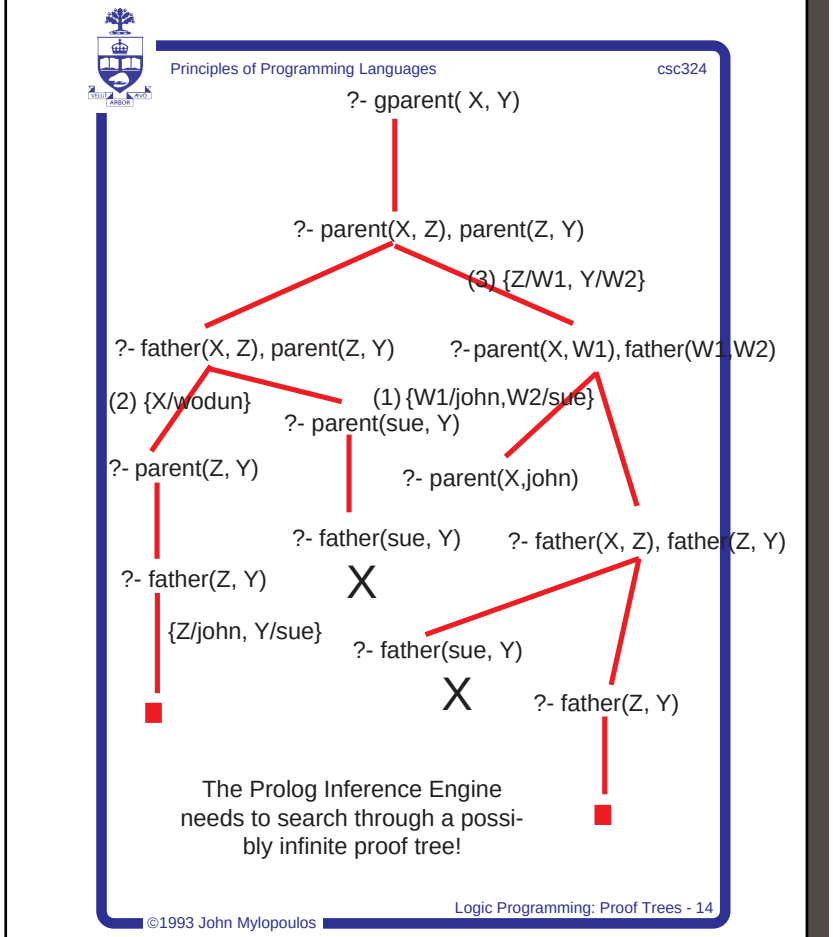
13



What does the user type in?

```
one.cdf% cprolog
C-PROLOG version 1.4
| ?- [user].        % user enters
                    % "assert fact" mode
| father(john, sue).
| father(wodun, X).
| parent(X, Y) :- father(X, Y).
| gparent(X, Z)
     :- parent(X,Y),parent(Y, Z).
| ^D                % user exits "assert
                    % fact" mode
yes
| ?- father(john, sue).    % user's goal
yes                 % Prolog's answer
| ?- father(X, sue).     % another goal
X = john;           % an answer; user
                    % asks for more ";"
X = wodun;          % a second answer;
                    % any more?
no                  % no more answers
```


14



The Prolog Inference Engine needs to search through a possibly infinite proof tree!

15

Variables - Bound, or Unbound During a Proof

Once a variable has been substituted by a term during a proof, it is bound to that term; otherwise a variable is free

At any one time during a proof, a variable is

- free, in which case it may be bound through unification, or
- bound to another variable, or
- bound to a constant, and has a value, or
- it is bound to a term, whose elements may or may not be bound, and has the term as its value.

©1993 John Mylopoulos

16

Unifying Two Variables

To unify two variables X and Y during a proof, do the following:

- Dereference X and Y:
  - Do until X is unbound, or X has a value:
    replace X with the binding of X, bind(X)
  - Do until Y is unbound, or Y has a value:
    replace Y with the binding of Y, bind(Y)
- Call MGU(bind(X), bind(Y))

For example, suppose
X ← W ← a
Y ← Z ← Z1 ← V
then derefencing will result in the evaluation of MGU(a,V)

©1993 John Mylopoulos