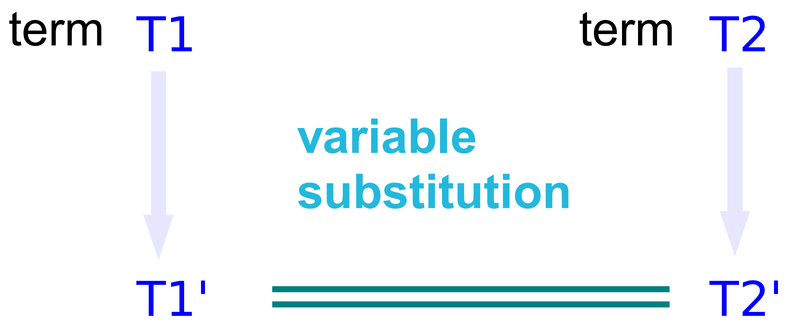
Unification (7A)

Young W. Lim
4/23/14

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

Unification



term T1

term T2

variable substitution

T1'

T2'

?- p(X, f(Y), a) = p(a, f(a), Y).

{X/a, Y/a}

p(a, f(a), a) = p(a, f(a), a)

```
?- p(X,f(Y),a) = p(a,f(a),Y).
X = a  Y = a

?- p(X,f(Y),a) = p(a,f(b),Y).
No
```

?- p(X, f(Y), a) = p(a, f(b), Y).

{X/a, ~~Y/b, Y/a~~}

Sharing References

```
?- p(X, f(Y), a) = p(Z, f(b), a).
X = _G182   Y = b   Z = _G182
```

```
?- p(X, f(Y), a) = p(Z, f(b), a).
```

{X/_G182, Y/b, Z/_G182}

p(_G182, f(b), a)

```
?- p(X,f(Y),a) = p(Z,f(b),a), X is d.
X = d   Y = b   Z = d
```

Operators: (=) and (is)

?Term1 **=** ?Term2

Unify Term1 with Term2.

=(Term, Term).

-Number **is** +Expr

True when Number is the value to which Expr evaluates. Typically, is/2 should be used with unbound left operand. If equality is to be tested, **=:=**/2 should be used.

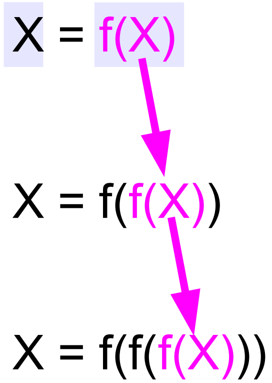
```
?- 1 is sin(pi/2).       Fails! sin(pi/2) evaluates to the float 1.0,
                         which does not unify with the integer 1.
?- 1 =:= sin(pi/2).      Succeeds as expected.
```

Occur Check

Prolog does not perform an occurs check

The circular reference : **

```
?- X=f(X).
X = f(**)
```



X = f(X)

X = f(f(X))

X = f(f(f(X)))

this goal succeeds with {X/f(f(f(...)))}

to break the circular reference

```
?- X=f(X), X=a.
No
```

or

```
?- X \= f(_).
Yes
```

\= cannot be unified with

_ (underscore): a wild card
can match anything

1. term1 & term2 : constants,
   unify *iff* they are the same atom or the same number

2. term1 : a variable, term2: any type of term,
   unify and term1 is instantiated to term2
   term1 : any type of term, term2: a variable,
   unify and term2 is instantiated to term1
   term1 & term2 : both variales
   unify and both are instantiated to each other (share values)

3 term1 & term2 : complex terms,
   unify *iff* they have the same functor and arity, and
   all their corresponding arguments unify, and
   the variable instantiations are compatible.

loves(vincent,X)
loves(X,mia)

4 Two terms
   unify *iff* it follows from the previous three clauses that they unify.

The Herbrand Unification Algorithm

```
Initialization step
    Initialize $\sigma$ to {} Initialize failure to false
    Push the equation $T1 = T2$ on the stack
Loop
    repeat {
        pop $x = y$ from the stack
        if x is a constant and $x == y$. Continue.
        else if x is a variable and x does not appear in y.
            Replace x with y in the stack and in $\sigma$. Add the substitution $\{x = y\}$ to $\sigma$.
        else if x is a variable and $x == y$. Continue.
        else if y is a variable and x is not a variable.
            Push $y = x$ on the stack.
        else if x and y are compounds with $x = f(x1, ..., xn)$ and $y = f(y1, ..., yn)$.
            Push on the stack $xi = yi$ for i ranging from 1 to n.
        else Set failure to true, and $\sigma$ to {}. Break.
} until (stack $\neq \emptyset$)
```

References

[1] en.wikipedia.org
[2] en.wiktionary.org
[3] U. Endriss, “Lecture Notes : Introduction to Prolog Programming”
[4] http://www.learnprolognow.org/ Learn Prolog Now!
[5] http://www.csupomona.edu/~jrfisher/www/prolog_tutorial
[6] www.cse.unsw.edu.au/~billw/cs9414/notes/prolog/intro.html
[7] www.cse.unsw.edu.au/~billw/dictionaries/prolog/negation.html
[8] http://ilppp.cs.lth.se/, P. Nugues, An Intro to Lang Processing with Perl and Prolog