Ex2:

S1 = {A/?x, B/?y, C/?w, D/?z}

S2 = {g(?x, ?y)/?z}

S1S2 = {A/?x, B/?y, C/?w, D/?z}

Change the order of the substitutions in the previous example.

S2S1 = { A/?x, B/?y, C/?w, g(A, B)/?z}

S1S2 $\neq$ S2S1

Verifying Consistency of Substitutions

To determine consistency, create two sets, T, V. Put all terms for all involved substitutions in T, and all variables for all involved substitutions in V. Then propagate values of the variables. If any variable has to take more than one constant value, then substitutions are inconsistent.

<u>Ex:</u> We have the following substitutions:

S1 = {A/?x, A/?y}
S2 = {A/?z. ?z/?x}

$$\begin{tikzcd}
 & & & A \\
T = \{ A \arrow[d] & A \arrow[d] & A \arrow[d] & \cancel{?z}\} \arrow[d] \\
V = \{?x & ?y & ?z \arrow[ur] & \cancel{?x}\} \\
 & & & A
\end{tikzcd}$$

Consistent

Using B/?z in S2:

$$\begin{tikzcd}
 & & & B \\
T = \{ A \arrow[d] & A \arrow[d] & B \arrow[d] & \cancel{?z}\} \arrow[d] \\
V = \{?x & ?y & ?z \arrow[ur] & \cancel{?x}\} \\
 & & & A
\end{tikzcd}$$

Inconsistent

How do we find substitutions?

$$\left.\begin{array}{c} W1(?x) \Rightarrow W2(?x) \\ W1(A) \end{array}\right\} \Rightarrow W2(A)$$

Use the recursive Unify procedure for unifying formulas (wffs) and finding substitutions.

Recursive Procedure **UNIFY**(E1, E2)

```
if either E1 or E2 is an atom (that is, a predicate symbol, a function
symbol, a constant symbol, a negation symbol or a variable),
interchange the arguments E1 and E2 (is necessary) so that E1 is an
atom, and do:
    begin
        if E1 and E2 are identical, return NIL
        if E1 is a variable, do:
            begin
                if E1 occurs in E2, return FAIL
                return {E2/E1}
            end
        if E2 is a variable, return {E1/E2}
        return FAIL
    end
```

contd…

…contd

F1 <- the first element of E1, T1 <- the rest of E1

F2 <- the first element of E2, T2 <- the rest of E2

Z1 <- **UNIFY**(F1,F2)

if Z1 = FAIL, **return** FAIL

G1 <- result of applying Z1 to T1

G2 <- result of applying Z1 to T2

Z2 <- **UNIFY**(G1, G2)

if Z2 = FAIL, **return** FAIL

return the composition of Z1 and Z2

Ex: {P[A, ?x, f(?y)], P[?y, B, f(?z)]}

First convert everything into lists.

E1 = (P A ?x (f ?y))
E2 = (P ?y B (f ?z))

F1 = P	T1 = (A ?x (f ?y))
F2 = P	T2 = (?y B (f ?z))

Z1 ← UNIFY(F1, F2) → return NIL

G1 = T1 = (A ?x (f ?y))
G2 = T2 = (?y B (f ?z))

Z2 ← UNIFY ((A ?x (f ?y)) (?y B (f ?z)))
E1 E2

F1 = A T1 = (?x (f ?y))
F2 = ?y T2 = (B (f ?z))

Z1 ← UNIFY (A, ?y) →return (A/?y)
F1 F2

G1 = (?x (f A))
G2 = (B (f ?z))

Z2 ← UNIFY ((?x (f A)) (B (f ?z)))

E1 E2

F1 = ?x T1 = ((f A))

F2 = B T2 = ((f ?z))

Z1 ← UNIFY(?x B) → return (B/?x)

Z2 ← UNIFY(((f A)) ((f ?z)))

:

Z2 ← UNIFY((A) (?z))

F1 = A T1 = ( )

F2 = ?z T2 = ( )

Z1 ← UNIFY (A ?z) → return (A/?z)

Compose ((A/?z) (B/?x))

$\downarrow$

((A/?z, B/?x))

Compose ((A/?y) (A/?z, B/?x))

$\downarrow$

((A/?y, A/?z, B/?x))

Compose (NIL (A/?y, A/?z, B/?x))

$\downarrow$

((A/?y, A/?z, B/?x))

Rule Systems: Simplest Knowledge Systems

Rules
$$\left\{\begin{array}{l} W1(v_1, v_2\ldots) \Rightarrow W1'(\ldots) \\ W2(\ldots) \Rightarrow W2'(\ldots) \\ : \\ : \\ Wn(\ldots) \Rightarrow Wn'(\ldots) \end{array}\right.$$

W1'(…) S_i

newly generated fact

$S_i = \{C_1/v_1, C_2/v_2, \ldots\}$

Facts
$$\left\{\begin{array}{l} W_{f1}\ (C1, C2, \ldots) \\ W_{f2}\ (C1, C2, \ldots) \\ : \end{array}\right.$$

Forward Chaining: match fact expressions with antecedents, generate new facts.

Backward Chaining: match goal or subgoal expressions with consequents, generate new subgoals.

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 **in** *KB* such that for some *I*,
UNIFY (p_i, p) = αι succeeds **do**
    FIND-AND-INFER (*KB*, [p_i, …, p_{i-1}, p_{i+1}, …, p_n], q, θ)
end

procedure FIND-AND-INFER (*KB*, *premises*,
    conclusions,θ)

if *premises* = [ ] **then**
  FORWARD-CHAIN (KB, SUBST (θ, *conclusion*))
else for each *p'* **in** KB such that UNIFY (*p*', SUBST(θ ,
FIRST (*premises*))) = θ_2 **do**
    FIND-AND-INFER (KB, REST (*premises*), *conclusion*,
    COMPOSER (θ, θ_2))
end

<u>Figure</u> - The forward-chaining inference algorithm. It adds to *KB* all the sentences that can be inferred from the sentence *p*. If *p* is already in *KB*, it does nothing. If *p* is new, consider each implication that has a premise that matches *p*. For each such implication, if all the remaining premises are in *KB*, then infer the conclusion. If the premises can be matched several ways, then infer each corresponding conclusion. The substitution of θ keeps track of the way things match.

Review and Exercises

- The following expressions are given:

  Manager(PURCHASING-DEPT, JOHN-JONES)
  Works-in(PURCHASING-DEPT, JOE-SMITH)

We also have the following production:

$$[\text{Works-in}(?x, ?y) \wedge \text{Manager}(?x, ?z)]$$
$$\Rightarrow \text{Boss-of}(?y, ?z)$$

a) Using these expressions, show the forward inference steps which show that JOHN-JONES is the boss of JOE-SMITH.

b) Write a goal for finding the boss of JOE-SMITH. Using the above expressions, by setting up subgoals through backward inference steps and matching show JOHN-JONES is his boss.

1a)

$$\left.\begin{array}{ll} W_{f1} & \text{Manager(PURCHASING-DEPT, JOHN-JONES)} \\ W_{f2} & \text{Works-in(PURCHASING-DEPT, JOE-SMITH)} \end{array}\right\} \text{F}$$

$$\left.\begin{array}{ll} W_{R1} & [\text{Works-in}(?x, ?y) \wedge \text{Manager}(?x, ?z)] \\ & \Rightarrow \underbrace{\text{Boss-of}(?y, ?z)}_{\text{W'}} \end{array}\right\} \text{R}$$

$$\left.\begin{array}{l} \text{S = \{PURCHASING-DEPT/?x} \\ \quad\quad \text{JOE-SMITH/?y,} \\ \quad\quad \text{JOHN-JONES /?z\}} \end{array}\right\} \begin{array}{l} \text{Result} \\ \text{from} \\ \text{Alg. Unify} \end{array}$$

Add to database W's

Boss-of (?y, ?z) S =

W_{f3} $\underbrace{\text{Boss-of(JOE-SMITH, JOHN-JONES)}}_{\text{goal}}$

1b)

goal: Boss-of(JOE-SMITH, ?t)

S2 = {JOE-SMITH /?y, ?t /?z}

New subgoals:

$$\left\{\begin{array}{l}\text{Works-in(?x, ?y) S2}\\ \text{Manager(?x, ?z) S2}\end{array}\right.$$

$$\downarrow$$

$$\left\{\begin{array}{ll}\text{Works-in(?x, JOE-SMITH)} & \text{sg1}\\ \text{Manager(?x, ?t)} & \text{sg2}\end{array}\right.$$

Match with facts in database:

S3 = {PURCHASING-DEPT/?x}

Works-in(PURCHASING-DEPT, JOE-SMITH)

sg1 S3 = W_{f2}

S4 = {PURCHASING-DEPT/?x, JOHN-JONES /?t}

sg2 S4 = W_{f1}

Compose S2, S3, S4 and check for consistency.

T = {JOE-SMITH ?t PURCHASING-DEPT …*

V = { ?y ?z ?x …**

*… PURCHASING –DEPT JOHN-JONES}

**… ?x ?t }

Compose S2, S3, S4 and check for consistency.

JOHN-JONES

T = {JOE-SMITH ~~?t~~ PURCHASING-DEPT …*

↓ ↓ ↓

V = { ~~?y~~ ~~?z~~ ~~?x~~ …**

JOE-SMITH PURCHASING-DEPT

JOHN-JONES

*… PURCHASING –DEPT JOHN-JONES}

↓ ↕

**… ~~?x~~ ?t }

PURCHASING-DEPT

Consistent substitutions Answer: John-Jones/?t