Composition

• If S and S’ are two substitutions sets, then the composition of S and S’ (written as SS’) is obtained by applying S’ to the elements of S and adding the result to S.

Example

• in the sequence of substitutions
  \{X/Y,W/Z\},\{V/X\},\{a/V,f(b)/W\}, composition will be
  \{a/Y,f(b)/Z\}

  because \{X/Y,W/Z\} with \{V/X\} to yield \{V/Y,W/Z\} and
  composing this with \{a/V,f(b)/W\} produces \{a/Y,f(b)/Z\}

  • composition can be shown to be associative but not commutative
Most general unifier

• Unifier be as general as possible that the most general
  unifier be found for the two expressions
• if generality is lost in the solution process it will lessen
  the scope of the eventual solution or even eliminate the
  possibility of a solution entirely
• Example
• in unifying p(X) and p(Y), any constant expression such
  as {fred/X,fred/Y} will do the trick, but not most general
• any variable would work such as {Z/X,Z/Y}
• fred would be a unifier, but it would lessen the generality
  of the result
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 $E$.

• most general unifier for a set of expression is unique except for the alphabetic variations
• That is, whether a variable is eventually called $X$ or $Y$ really does not make any difference to the generality of the resulting unifications
• unification is important for any artificial intelligence problem solver that uses the predicate calculus for representation
• unification specifies conditions under which two or more predicate calculus expressions may be said to be equivalent
• allow use of inference rules such as resolution, a process that often requires backtracking to find all possible interpretations
• Unify algorithm assumes a slightly modified syntax

<table>
<thead>
<tr>
<th>Predicate Calculus Syntax</th>
<th>List Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>p(a,b)</td>
<td>(p a b)</td>
</tr>
<tr>
<td>p(f(a), g(X,Y))</td>
<td>(p (f a) (g X Y))</td>
</tr>
<tr>
<td>equal(eve, mother(cain))</td>
<td>(equal eve (mother cain))</td>
</tr>
</tbody>
</table>
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:
                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
Example of unification

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

• first
  unify(parents, parents)

• second
  unify(X,bill) return {bill/X}

• third
  unify((father bill) (mother bill)), ((father bill) Y))
unify((father bill), (father bill))
unify(father,father)
unify(bill,bill)
unify(( ),( ))

unify((mother bill),(Y)) return {(mother bill) / Y}
unify((mother bill, Y)
unify(( ),( ))

After composing
the answer becomes {bill / X , (mother bill) / Y}
Figure 2.5: Further steps in the unification of (parents X (father X) (mother bill)) and (parents bill (father bill) Y)

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

   Unify first elements and apply substitutions to rest

   return {}

2. unify(parents, parents)

3. unify(((X (father X) (mother bill)),(bill (father bill) Y)))

   Unify first elements and apply substitutions to rest

   return {bill/X}

4. unify(X,bill)

5. unify(((father bill) (mother bill)),((father bill) Y))
Figure 2.6: Final trace of the unification of (parents X (father X) (mother bill)) and (parents bill (father bill) Y).
Unify first elements and apply substitutions to rest

return {}  

6. unify((father bill), (father bill))

Unify first elements and apply substitutions to rest  

return {}

7. unify(father, father)

8. unify((bill), (bill))

Unify first elements and apply substitutions to rest

return {}

9. unify(bill, bill)

10. unify((), ())

11. unify((mother bill)), (Y))

Unify first elements and apply substitutions to rest  

return {}

12. unify((mother bill), Y)

13. unify((), ())

return {(mother bill)/Y}
Application:
A Logic-Based Financial Advisor

• To help a user decide whether to invest in a savings account or the stock market. Some investors may want to split their money between the two. The investment that will be recommended for individual investors depends on their income and the current amount they have saved according to the following criteria:

• individual with an inadequate saving account should always make increasing the amount saved their first priority, regardless of their income

• individuals with an adequate savings account and an adequate income should consider a riskier but potentially more profitable investment in the stock market

• individuals with a lower income who already have an adequate savings account may want to consider splitting their surplus income between savings and stocks, to increase the cushion in saving while attempting to increase their income through stocks
A Logic-Based Financial Advisor

Suppose

- adequacy of both savings and income is determined by the number of dependents an individual must support

- at least $5,000 in the bank for each dependent

- an adequate income must be a steady income and supply at least $15,000 per year plus an additional $4,000 for each dependent
In predicate calculus

- \texttt{savings\_account} for adequacy savings
- \texttt{income} for adequacy income
  with argument adequate and inadequate

\texttt{savings\_account(adequate)}.  
\texttt{savings\_account(inadequate)}.  
\texttt{income(adequate)}.  
\texttt{income(inadequate)}.  

- For conclusion, unary predicate \texttt{investment} with possible arguments \texttt{stocks}, \texttt{savings}, or \texttt{combination}
A Logic-Based Financial Advisor rules

- The first rule that individuals with inadequate saving should make increased saving their main priority

\[
\text{savings\_account(adequate)} \rightarrow \text{investment(savings)}
\]

- Similarly, the remaining two possible investment alternatives are

\[
\text{savings\_account(adequate)} \land \text{income(adequate)} \rightarrow \text{investment(stocks)}
\]

\[
\text{savings\_account(adequate)} \land \text{income(inadequate)} \rightarrow \text{investment(combination)}
\]
More rules

• To determine the minimum adequate savings, the function minsaving is defined with one argument, the number of dependent, and returns 5000 times the argument, such as

\[ \text{minsavings}(X) \equiv 5000 \times X \]

• adequacy of saving is defined by rules

\[
\forall X \, \text{amount}_\text{saved}(X) \land \exists Y (\text{dependents}(Y) \land \text{greater}(X, \text{min savings}(Y))) \Rightarrow \text{savings}_\text{account}(\text{adequate})
\]

\[
\forall X \, \text{amount}_\text{saved}(X) \land \exists Y (\text{dependents}(Y) \land \neg \text{greater}(X, \text{min savings}(Y))) \Rightarrow \text{savings}_\text{account}(\text{inadequate})
\]
More rules continued

• Similarly a function minincome is defined

\[\text{minincome}(X) \equiv 15000 + (4000 \times X)\]

• investor’s income is represented by a predicate \textit{earnings} with two arguments amount earned and steady or unsteady

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

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

\[
\forall X \, \text{earnings}(X, \text{unsteady}) \rightarrow \text{income}(\text{inadequate})
\]
Example

• for an individual with three dependents, $22,000 in savings, and a steady income of 25,000 would be

\begin{verbatim}
amount_saved(22000)
earnings(25000,steady)
dependent(3)
\end{verbatim}
1. savings_account(inadequate) $\rightarrow$ investment(savings).

2. savings_account(adequate) $\land$ income(adequate) $\rightarrow$ investment(stocks).

3. savings_account(adequate) $\land$ income(inadequate) $\rightarrow$ investment(combination).

4. $\forall$ amount_saved(X) $\land$ $\exists$ Y (dependents(Y) $\land$
   greater(X, minsavings(Y))) $\rightarrow$ savings_account(adequate).

5. $\forall$ X amount_saved(X) $\land$ $\exists$ Y (dependents(Y) $\land$
   $\neg$ greater(X, minsavings(Y))) $\rightarrow$ savings_account(inadequate).

6. $\forall$ X earnings(X, steady) $\land$ $\exists$ Y (dependents(Y) $\land$
   greater(X, minincome(Y))) $\rightarrow$ income(adequate).

7. $\forall$ X earnings(X, steady) $\land$ $\exists$ Y (dependents(Y) $\land$
   $\neg$ 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).
Unifying and inferring

• To unify the conjunction of (10) and (11) with the first components of the premise of 7

\[ \forall X \, \text{earnings}(X, \text{steady}) \land \exists Y \, (\text{dependents}(Y) \land \neg \text{greater}(X, \min income(Y))) \]\n\[ \rightarrow \text{income (inadequate)} \]

i.e. \[ \text{earnings}(25000, \text{steady}) \land \text{dependents}(3) \] unifies with
\[ \text{earnings}(X, \text{steady}) \land \text{dependents}(Y) \] under the substitution
\{25000/X, 3/Y\}

• This substitution yields

\[ \text{earnings}(25000, \text{steady}) \land \text{dependents}(3) \land \neg \text{greater}(25000, \min income(3)) \]\n\[ \rightarrow \text{income (inadequate)} \]
Unifying and inferring continued

• Since dependent is 3, $minincome(3)$ is 27000, so

$earnings(25000, steady) \land dependents (3) \land \neg greater (25000, 27000) \rightarrow income (inadequate)$

• by applying modus ponens, the conclusion becomes $income(inadequate)$

• This is added as assertion 12

$12. income(inadequate)$
Unifying and inferring continued

• Similarly,

\[ amount \_\text{saved} (22000) \land dependent \ (3) \] unifies with the first two elements of the premise of assertion 4 under the substitutions \{22000/X,3/Y\} yielding the implication

\[ amount \_\text{saved} (22000) \land dependent \ (3) \land greater (22000,\text{minsavings} (3)) \]
\[ \rightarrow \text{savings} \_\text{account}(\text{adequate}) \]

• since \text{minsavings}(3) is 15000

\[ amount \_\text{saved} (22000) \land dependent \ (3) \land greater (22000,15000) \]
\[ \rightarrow \text{savings} \_\text{account}(\text{adequate}) \]

• applying modus ponens, conclusion becomes

\[ \text{saving} \_\text{account}(\text{adequate}) \]
Final conclusion

• This conclusion is added as assertion 13,
  \[13. \text{savings\_account}(\text{adequate})\]

• As an examination of expressions 3, 12, and 13 indicates, the premise of implication 3 is also true
  \[\text{savings\_account}(\text{adequate}) \land \text{income}(\text{inadequate}) \rightarrow \text{investment}(\text{combination})\]

• Therefore, applying modus ponens again and final conclusion becomes
  \[\text{investment}(\text{combination})\]