Reasoning with Variables

- An instance of an atom or a clause is obtained by uniformly substituting terms for variables.
- A substitution is a finite set of the form $\{V_1/t_1, \ldots, V_n/t_n\}$, where each V_i is a distinct variable and each t_i is a term.
- The application of a substitution $\sigma = \{V_1/t_1, \ldots, V_n/t_n\}$ to an atom or clause e, written $e\sigma$, is the instance of e with every occurrence of V_i replaced by t_i.

Application Examples

The following are substitutions:

- $\sigma_1 = \{X/A, Y/b, Z/C, D/e\}$
- $\sigma_2 = \{A/X, Y/b, C/Z, D/e\}$
- $\sigma_3 = \{A/V, X/V, Y/b, C/W, Z/W, D/e\}$

The following shows some applications:

- $p(A, b, C, D)\sigma_1 =$
- $p(X, Y, Z, e)\sigma_1 =$
- $p(A, b, C, D)\sigma_2 =$
- $p(X, Y, Z, e)\sigma_2 =$
- $p(A, b, C, D)\sigma_3 =$
- $p(X, Y, Z, e)\sigma_3 =$

Application Examples

The following are substitutions:

- $\sigma_1 = \{X/A, Y/b, Z/C, D/e\}$
- $\sigma_2 = \{A/X, Y/b, C/Z, D/e\}$
- $\sigma_3 = \{A/V, X/V, Y/b, C/W, Z/W, D/e\}$

The following shows some applications:

- $p(A, b, C, D)\sigma_1 = p(A, b, C, e)$
- $p(X, Y, Z, e)\sigma_1 = p(A, b, C, e)$
- $p(A, b, C, D)\sigma_2 = p(X, b, Z, e)$
- $p(X, Y, Z, e)\sigma_2 = p(X, b, Z, e)$
- $p(A, b, C, D)\sigma_3 = p(V, b, W, e)$
- $p(X, Y, Z, e)\sigma_3 = p(V, b, W, e)$

Unifiers

- Substitution σ is a **unifier** of e_1 and e_2 if $e_1\sigma = e_2\sigma$.
- Substitution σ is a **most general unifier** (mgu) of e_1 and e_2 if
  - σ is a unifier of e_1 and e_2; and
  - if substitution σ' also unifies e_1 and e_2, then $e\sigma'$ is an instance of $e\sigma$ for all atoms e.
- If two atoms have a unifier, they have a most general unifier.

Unification Example

Which of the following are unifiers of $p(A, b, C, D)$ and $p(X, Y, Z, e)$:

- $\sigma_1 = \{X/A, Y/b, Z/C, D/e\}$
- $\sigma_2 = \{Y/b, D/e\}$
- $\sigma_3 = \{X/A, Y/b, Z/C, D/e, W/a\}$
- $\sigma_4 = \{A/X, Y/b, C/Z, D/e\}$
- $\sigma_5 = \{X/a, Y/b, Z/c, D/e\}$
- $\sigma_6 = \{A/a, X/a, Y/b, C/c, Z/c, D/e\}$
- $\sigma_7 = \{A/V, X/V, Y/b, C/W, Z/W, D/e\}$
- $\sigma_8 = \{X/A, Y/b, Z/A, C/A, D/e\}$

Which are most general unifiers?

Unification Example

$p(A, b, C, D)$ and $p(X, Y, Z, e)$ have as unifiers:

- $\sigma_1 = \{X/A, Y/b, Z/C, D/e\}$
- $\sigma_4 = \{A/X, Y/b, C/Z, D/e\}$
- $\sigma_7 = \{A/V, X/V, Y/b, C/W, Z/W, D/e\}$
- $\sigma_6 = \{A/a, X/a, Y/b, C/c, Z/c, D/e\}$
- $\sigma_8 = \{X/A, Y/b, Z/A, C/A, D/e\}$
- $\sigma_3 = \{X/A, Y/b, Z/C, D/e, W/a\}$

The first three are most general unifiers.
The following substitutions are not unifiers:

- $\sigma_2 = \{Y/b, D/e\}$
- $\sigma_5 = \{X/a, Y/b, Z/c, D/e\}$

Proofs

- A proof is a mechanically derivable demonstration that a formula logically follows from a knowledge base.
- Given a proof procedure, $KB \vdash g$ means g can be derived from knowledge base KB.
- Recall $KB \models g$ means g is true in all models of KB.
- A proof procedure is sound if $KB \vdash g$ implies $KB \models g$.
- A proof procedure is complete if $KB \models g$ implies $KB \vdash g$.

Bottom-up proof procedure

$KB \vdash g$ if there is g' added to C in this procedure where $g = g'\theta$:

$C := \{\}$;
repeat
  select clause "$h \leftarrow b_1 \wedge \ldots \wedge b_m$" in KB such that
    there is a substitution θ such that
    for all i, there exists $b'_i \in C$ where $b_i\theta = b'_i\theta$ and
    there is no $h' \in C$ such that h' is more general than $h\theta$
  $C := C \cup \{h\theta\}$
until no more clauses can be selected.

Example

$$live(Y) \leftarrow connected\_to(Y, Z) \wedge live(Z). \quad live(outside).$$
$$connected\_to(w_6, w_5). \quad connected\_to(w_5, outside).$$

Example

$live(Y) \leftarrow connected\_to(Y, Z) \wedge live(Z). \;\; live(outside).$

$connected\_to(w_6, w_5). \quad connected\_to(w_5, outside).$

$$
\begin{aligned}
C = \{ & live(outside), \\
& connected\_to(w_6, w_5), \\
& connected\_to(w_5, outside), \\
& live(w_5), \\
& live(w_6)\}
\end{aligned}
$$

Soundness of bottom-up proof procedure

If $KB \vdash g$ then $KB \models g$.

- Suppose there is a g such that $KB \vdash g$ and $KB \not\models g$.
- Then there must be a first atom added to C that has an instance that isn't true in every model of KB. Call it h. Suppose h isn't true in model I of KB.
- There must be a clause in KB of form

  $$h' \leftarrow b_1 \wedge \ldots \wedge b_m$$

  where $h = h'\theta$. Each b_i is true in I. h is false in I. So this clause is false in I. Therefore I isn't a model of KB.
- Contradiction.

Fixed Point

- The C generated by the bottom-up algorithm is called a fixed point.
- C can be infinite; we require the selection to be fair.
- Herbrand interpretation: The domain is the set of constants.
  We invent one if the KB or query doesn't contain one.
  Each constant denotes itself.
- Let I be the Herbrand interpretation in which every ground instance of every element of the fixed point is true and every other atom is false.
- I is a model of KB.
  Proof: suppose $h \leftarrow b_1 \wedge \ldots \wedge b_m$ in KB is false in I. Then h is false and each b_i is true in I. Thus h can be added to C. Contradiction to C being the fixed point.
- I is called a Minimal Model.

Completeness

If $KB \models g$ then $KB \vdash g$.

- Suppose $KB \models g$. Then g is true in all models of KB.
- Thus g is true in the minimal model.
- Thus g is in the fixed point.
- Thus g is generated by the bottom up algorithm.
- Thus $KB \vdash g$.

Top-down Proof procedure

- A generalized answer clause is of the form

  $$yes(t_1, \ldots, t_k) \leftarrow a_1 \wedge a_2 \wedge \ldots \wedge a_m,$$

  where $t_1, \ldots, t_k$ are terms and $a_1, \ldots, a_m$ are atoms.
- The SLD resolution of this generalized answer clause on a_i with the clause

  $$a \leftarrow b_1 \wedge \ldots \wedge b_p,$$

  where a_i and a have most general unifier θ, is

  $$(yes(t_1, \ldots, t_k) \leftarrow$$
  $$a_1 \wedge \ldots \wedge a_{i-1} \wedge b_1 \wedge \ldots \wedge b_p \wedge a_{i+1} \wedge \ldots \wedge a_m)\theta.$$

To solve query $?B$ with variables $V_1, \ldots, V_k$:

Set ac to generalized answer clause $yes(V_1, \ldots, V_k) \leftarrow B$;
While ac is not an answer **do**

Suppose ac is $yes(t_1, \ldots, t_k) \leftarrow a_1 \land a_2 \land \ldots \land a_m$
Select atom a_i in the body of ac;
Choose clause $a \leftarrow b_1 \land \ldots \land b_p$ in KB;
Rename all variables in $a \leftarrow b_1 \land \ldots \land b_p$;
Let θ be the most general unifier of a_i and a.
Fail if they don't unify;
Set ac to $(yes(t_1, \ldots, t_k) \leftarrow a_1 \land \ldots \land a_{i-1} \land$
$b_1 \land \ldots \land b_p \land a_{i+1} \land \ldots \land a_m)\theta$

end while.

Example

$live(Y) \leftarrow connected\_to(Y, Z) \wedge live(Z).$ $live(outside).$

$connected\_to(w_6, w_5).$ $connected\_to(w_5, outside).$

$?live(A).$

Example

$live(Y) \leftarrow connected\_to(Y, Z) \wedge live(Z).$ $live(outside).$

$connected\_to(w_6, w_5).$ $connected\_to(w_5, outside).$

$?live(A).$

$$yes(A) \leftarrow live(A).$$
$$yes(A) \leftarrow connected\_to(A, Z_1) \wedge live(Z_1).$$
$$yes(w_6) \leftarrow live(w_5).$$
$$yes(w_6) \leftarrow connected\_to(w_5, Z_2) \wedge live(Z_2).$$
$$yes(w_6) \leftarrow live(outside).$$
$$yes(w_6) \leftarrow .$$

```
Procedure Unify(t1,t2)
        Inputs:
                t1,t2: atoms Output
        Output: most general unifier of t1 and t2 if it exists or ⊥ otherwise
        Local
                E: a set of equality statements
                S: substitution
        E ←{t1=t2}
        S={}
          while (E≠{})
                  select and remove x=y from E
                  if (y is not identical to x) then
                          if (x is a variable) then
                                  replace x with y everywhere in E and S
                                  S←{x/y}∪S
                          else if (y is a variable) then
                                  replace y with x everywhere in E and S
                                  S←{y/x}∪S
                          else if (x is f(x1,...,xn) and y is f(y1,...,yn)) then
                                  E←E∪{x1=y1,...,xn=yn}
                          else
                                  return ⊥
          return S
```

Example Suppose we want to unify p(X,Y,Y) with p(a,Z,b). Initially E is {p(X,Y,Y)=p(a,Z,b)}. The first time through the while loop, E becomes {X=a,Y=Z,Y=b}. Suppose X=a is selected next. Then S becomes {X/a} and E becomes {Y=Z,Y=b}. Suppose Y=Z is selected. Then Y is replaced by Z in S and E. S becomes {X/a,Y/Z} and E becomes {Z=b}. Finally Z=b is selected, Z is replaced by b, S becomes {X/a,Y/b,Z/b}, and E becomes empty. The substitution {X/a,Y/b,Z/b} is returned as an MGU.

Function Symbols

- Often we want to refer to individuals in terms of components.
- Examples: 4:55 p.m. English sentences. A classlist.
- We extend the notion of term. So that a term can be $f(t_1, \ldots, t_n)$ where f is a function symbol and the t_i are terms.
- In an interpretation and with a variable assignment, term $f(t_1, \ldots, t_n)$ denotes an individual in the domain.
- One function symbol and one constant can refer to infinitely many individuals.

Times during the day

- You can use the function symbol am so that am(H,M) denotes the time H:M a.m., when H is an integer between 1 and 12 and M is an integer between 0 and 59.
- For example, am(10,38) denotes the time 10:38 a.m.; am denotes a function from pairs of integers into times.
- Similarly, you can define the symbol pm to denote the times after noon.

```
before(am(H1,M1),pm(H2,M2)).
before(am(12,M1),am(H2,M2)) ←  H2<12.

before(am(H1,M1),am(H2,M2)) ←  H1<H2 ∧ H2<12.

before(am(H,M1),am(H,M2)) ←  M1<M2.

before(pm(12,M1),pm(H2,M2)) ←  H2<12.
```

```
before(pm(H1,M1),pm(H2,M2)) ← H1<H2 ∧ H2<12.
before(pm(H,M1),pm(H,M2)) ← M1<M2.
```

Lists

- A list is an ordered sequence of elements.
- Let's use the constant nil to denote the empty list, and the function $cons(H, T)$ to denote the list with first element H and rest-of-list T. These are not built-in.
- The list containing *sue*, *kim* and *randy* is

$$cons(sue, cons(kim, cons(randy, nil)))$$

- $append(X, Y, Z)$ is true if list Z contains the elements of X followed by the elements of Y

$$append(nil, Z, Z).$$

$$append(cons(A, X), Y, cons(A, Z)) \leftarrow append(X, Y, Z).$$

append(c(A,X),Y,c(A,Z)). ← append(X,Y,Z).

append(nil,Z,Z).

ask append(F,c(L,nil),c(l,c(i,c(s,c(t,nil))))).

yes(F,L)←append(F,c(L,nil),c(l,c(i,c(s,c(t,nil)))))

resolve with append(c(A1,X1),Y1,c(A1,Z1))←append(X1,Y1,Z1)

substitution: {F/c(l,X1), Y1/c(L,nil), A1/l ,Z1/c(i,c(s,c(t,nil)))}

yes(c(l,X1),L)←append(X1,c(L,nil),c(i,c(s,c(t,nil))))

resolve with append(c(A2,X2),Y2,c(A2,Z2))←append(X2,Y2,Z2)

substitution: {X1/c(i,X2), Y2/c(L,nil), A2/i ,Z2/c(s,c(t,nil))}

yes(c(l,c(i,X2)),L)←append(X2,c(L,nil),c(s,c(t,nil)))

resolve with append(c(A3,X3),Y3,c(A3,Z3))←append(X3,Y3,Z3)

substitution: {X2/c(s,X3), Y3/c(L,nil), A3/s ,Z3/c(t,nil)}

yes(c(l,c(i,c(s,X3))),L)←append(X3,c(L,nil),c(t,nil))

Both clauses are applicable. Choosing the first clause gives:

resolve with append(c(A4,X4),Y4,c(A4,Z4))←append(X4,Y4,Z4)

substitution: {X3/c(t,X4), Y4/c(L,nil), A4/t ,Z4/nil}

yes(c(l,c(i,c(s,X3))),L)←append(X4,c(L,nil),nil)

There are no clauses whose head unifies with the atom in the generalized answer clause's body. The proof fails.

Choosing the second clause instead of the first gives:

resolve with append(nil,Z5,Z5).

substitution: {Z5/c(t,nil),X3/nil,L/t}

yes(c(l,c(i,c(s,nil))),t)←

At this point, the proof succeeds, with answer F=c(l,c(i,c(s,nil))), L=t.

Using the list notation, append from the previous example can be written as:

```
append([A|X],Y,[A|Z])  ←  append(X,Y,Z).
append( [],Z,Z).
```

The query:

```
ask append(F,[L],[l,i,s,t])
```

has an answer `F=[l,i,s], L=t.`

The proof is exactly as in the previous example. As far as the proof procedure is concerned, nothing has changed; there is just a renamed function symbol and constant.