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:

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

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

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

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

The first three are most general unifiers.

The following substitutions are not unifiers:

- $\sigma_7 = \{Y/b, D/e\}$
- $\sigma_8 = \{X/a, Y/b, Z/c, D/e\}$

Bottom-up procedure

- You can carry out the bottom-up procedure on the ground instances of the clauses.
- Soundness is a direct corollary of the ground soundness.
- For completeness, we build a canonical minimal model. We need a denotation for constants:

  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.

Definite Resolution with Variables

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

Select atom a_i in the body of *ac*;

Choose clause $a \leftarrow b_1 \wedge \ldots \wedge b_p$ in *KB*;

Rename all variables in $a \leftarrow b_1 \wedge \ldots \wedge b_p$;

Let θ be the most general unifier of a_i and a.

Fail if they don't unify;

Set *ac* to $(yes(t_1, \ldots, t_k) \leftarrow a_1 \wedge \ldots \wedge a_{i-1} \wedge$

$b_1 \wedge \ldots \wedge b_p \wedge a_{i+1} \wedge \ldots \wedge a_m)\theta$

end while.

Example

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

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

$?live(A).$

$yes(A) \leftarrow live(A).$

$yes(A) \leftarrow connected\_to(A, Z_1) \wedge live(Z_1).$

$yes(w_6) \leftarrow live(w_5).$

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

$yes(w_6) \leftarrow live(outside).$

$yes(w_6) \leftarrow .$

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.

With one function symbol and one constant we can refer to infinitely many individuals.

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 $david$, $alan$ and $randy$ is

$$cons(david, cons(alan, cons(randy, nil)))$$

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

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

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