Terms

Definition 1. An expression t is a *term* if it is either

- a constant c,
- a variable X, or
- an expression $f(t_1, \ldots, t_n)$ where f is an n-place function symbol and $t_1, \ldots, t_n$ are terms.

In the Prolog literature, constants are often called 'atoms' and function symbols called 'functors'.

Substitution

Definition 2. An *assignment* is a pair s/X comprising a variable X and a term s which is taken to be the value for X.

A *substitution* σ is a finite list of assignments $[s_1/X_1, \ldots, s_n/X_n]$ subject to the conditions

- all of the variables $X_1, \ldots, X_n$ are distinct (that is, $X_i \neq X_j$ if $i \neq j$);
- none of the variables $X_1, \ldots, X_n$ appear in any of the terms $s_1, \ldots, s_n$.

If t is a term and $\sigma = [s_1/X_1, \ldots, s_n/X_n]$ is a substitution then we write $t\sigma$ for the term obtained by replacing each occurrence of a variable X_i in t by the corresponding term s_i.

The value of $t\sigma$ is calculated by recursion over the structure of the term t

$$
\begin{aligned}
c\sigma &= c \\
Y\sigma &= \begin{cases} s_i & \text{if } Y = X_i \\ Y & \text{if } Y \text{ does not appear in } X_1, \ldots, X_n \end{cases} \\
f(t_1, \ldots, t_m)\sigma &= f(t_1\sigma, \ldots, t_m\sigma)
\end{aligned}
$$

Example 3. If t is the term

$$appmt(date(D, M, Y), T)$$

and σ is the substitution

$$[23/D, june/M, 1912/Y, time(Hrs, Mins)/T]$$

then $t\sigma$ is the term

$$appmt(date(23, june, 1912), time(Hrs, Mins))$$

Example 4. Some substitutions on the term $f(X, h(X, Y, Z))$.

$$
\begin{array}{llcl}
f(X, h(X, Y, Z)) & [a/X, b/Y, c/Z] & = & f(a, h(a, b, c)) \\
f(X, h(X, Y, Z)) & [g(b)/X] & = & f(g(b), h(g(b), Y, Z)) \\
f(X, h(X, Y, Z)) & [A/X, f(A, b)/Y] & = & f(A, h(A, f(A, b), Z)) \\
f(X, h(X, Y, Z)) & [k(a, B)/W, c/V] & = & f(X, h(X, Y, Z))
\end{array}
$$

Formulae

Definition 5. If p is an n-place predicate symbol and $t_1, \ldots, t_n$ are terms then the expression $p(t_1, \ldots, t_n)$ is an *atomic formula*.

We shall denote atomic formulae by greek letters, typically ϕ, ψ and θ (*phi*, *psi* and *theta*).

Queries

Definition 6. A *query* or *goal* is a finite list of atomic formulae written

$$?\text{–}\ \phi_1, \ldots, \phi_n.$$

We write '$\square$' for the empty query.

Clauses

Definition 7. A *clause* is an expression of the form

$$\phi :- \psi_1, \ldots, \psi_n.$$

where ϕ and $\psi_1, \ldots, \psi_n$ are all atomic formulae. The formula ϕ on the left is known as the *head* of the clause, and the list $\psi_1, \ldots, \psi_n$ of formulae on the right is the *body*.

If there are no formulae in the body of the clause then the clause is known as a *unit clause*. We omit the :− symbol in a unit clause and write 'ϕ.' for short.

Logical meaning

A clause is essentially a statement in logic (universally quantified).

- "Everyone likes Sophie"

```
likes(X,sophie).
```

- "Sophie likes anyone who likes Colin"

```
likes(sophie,X):- likes(X,colin).
```

- "cats like anyone who strokes them and feeds them"

```
likes(X,Y):- cat(X), strokes(Y,X), feeds(Y,X).
```

The same statements might be written in the predicate calculus as

$$\forall X(likes(X, sophie))$$

$$\forall X(likes(X, colin) \rightarrow likes(sophie, X))$$

$$\forall X \forall Y(cat(X) \wedge strokes(Y, X) \wedge feeds(Y, X) \rightarrow likes(X, Y))$$

Example 8. Consider the following situation.

```
cat(colin).

human(mike).
human(steve).
human(sophie).

likes(mike,colin).
likes(steve,colin).
likes(X,Y):- cat(X), strokes(Y,X), feeds(Y,X).

strokes(Y,X):- cat(X), human(Y), likes(Y,X).

feeds(mike,colin).
feeds(sophie,colin).
```

Deduction

What can we deduce?

Well, Mike likes Colin so he must stroke him. He also feeds Colin so Colin must like Mike.

If we compile the program and try the query

```
?- likes(colin,Y).
```

then we obtain the solution

```
Y = mike
```

Prolog uses a form of automated deduction to establish that `likes(colin,mike)` holds true from the facts and rules available.

Logic Programming

The idea can be summed up:

program	=	a list of clauses Γ
query/goal	=	a list of atomic formulae ?– $\phi_1, \ldots, \phi_n$.
computation	=	search for a proof that $\phi_1, \ldots, \phi_n$ hold true
answer/result	=	a substitution σ such that the formulae $\phi_1\sigma, \ldots, \phi_n\sigma$ are all provable using the the clauses in Γ.

We say that σ is a computed *answer substitution* for the program Γ and query ?– $\phi_1, \ldots, \phi_n$.

Remark 9. It is often said that the variables in a query are *existentially* quantified. In setting the query

```
?- likes(colin,Y).
```

we are asking the question "does there *exist* a Y whom Colin likes". Any solution that appears is an answer that there does.

We might write the statement that there exists such a Y in the predicate calculus as

$$\exists Y(likes(colin, Y))$$