F9

Prolog

Lennart Andersson

Revision 2009-04-24

2009

Axioms

```
male(gustav).
feamale(stina).
feamale(eva).
feamale(lena).
father(gustav, eva).
father(gustav, lena).
father(oskar, gustav).
mother(stina, eva).
wife(gustav, stina).
```


Rules

```
husband(X, Y) :- wife(Y, X).
male(X) :- father(X, Y).
son(X, Y) :- male(X), father(Y, X).
sonOfGustav(X) :- male(X), father(gustav, X).
grandfather(X, Z) :- father(X, Y), father(Y, Z).
```


Queries

```
?- feamale(eva).
yes
?- feamale(anna).
no
?- male(oskar).
yes
```

Queries

```
?- grandfather(oskar, Y).
Y=eva
Y=lena
?- male(X).
X=gustav
X=oskar
```


Concrete grammar

$$
\begin{array}{lll}
\mathit{rule} & \rightarrow & \mathit{term}. \mid \mathit{term}\ \texttt{:-}\ \mathit{terms}. \\
\mathit{terms} & \rightarrow & \epsilon \mid \mathit{term}(,\mathit{term})^* \\
\mathit{term} & \rightarrow & c \mid X \mid f(\mathit{terms})
\end{array}
$$

Unification

```
Subst σ; // initialized to the empty substitutio
void unify(Term s, Term t) {
  if (s is a variable) s := σ s;
  if (t is a variable) t := σ t;
  if (s is a variable && s.equals(t)) {
    // do nothing
  } else if (s== f(s_1,...,s_n) and t== g(t_1,...,t_m)) {
    if (f.equals(g) and n == m) {
      for (int i=1; i≤ n; i++) {
        unify(s_i, t_i);
      }
    } else {
      exit with failure
    }
  } else ...
}
```


Unification

```
  } else if (s is not a variable) {
    unify(t, s);
  } else if (s occurs in t) {
    exit with failure
  } else {
    σ = σ[s ↦ t];
  }
}
```

Resolution

```
void resolve(RuleList rules, Term query) {
  TermList resolvent = [query];
  while (resolvent is not empty) {
    let t be the first term in resolvent
    choose a rule r = g :- terms
                  with fresh variable names such
      t and g unifies with most general unifier θ
      if there is no such rule
      then
        exit with failure
      else
        replace t in resolvent with terms.
      apply θ to resolvent
  }
}
```