VadeMeCum ProLog

Grammaire ProLog
https://www.complang.tuwien.ac.at/sicstus/sicstus_42.html

clause --> non-unit-clause | unit-clause

non-unit-clause --> head :- body

unit-clause --> head

head --> ~~module : head~~
| goal

body --> ~~module : body~~
| ~~body -> body ; body~~
| ~~body -> body~~
| ~~\+ body~~
| ~~body ; body~~
| body , body
| goal

goal --> term

term --> functor (arguments)
| ~~(subterm)~~
| ~~{ subterm }~~
| list
| string
| constant
| variable

arguments --> subterm
| subterm , arguments

subterm --> term

list --> []
| [listexpr]

listexpr --> subterm
| subterm , listexpr
| subterm | subterm

constant --> atom | number

atom --> name

functor --> name

Unification (Herbrand algorithm)
wwwdh.cs.fau.de/IMMD8/Lectures/LOGIK/isoprolog.pdf

Given a set of equations of the form $t_1 = t_2$ apply in any order one of the following non-exclusive steps:

a) If there is an equation of the form:

1) $f = g$ where f and g are different constants, or

2) $f = g$ where f is a constant and g is a compound term, or f is a compound term and g is a constant, or

3) $f(...) = g(...)$ where f and g are different functors, or

4) $f(a_1, a_2, ...a_N) = f(b_1, b_2, ...b_M)$ where N and M are different.

then exit with failure (*not unifiable*).

b) If there is an equation of the form $X = X$, X being a variable, then remove it.

c) If there is an equation of the form $c = c$, c being a constant, then remove it.

d) If there is an equation of the form $f(a_1, a_2, ...a_N) = f(b_1, b_2, ...b_N)$ then replace it by the set of equations $a_i = b_i$.

e) If there is an equation of the form $t = X$, X being a variable and t a non-variable term, then replace it by the equation $X = t$,

f) If there is an equation of the form $X = t$ where:

1) X is a variable and t a term in which the variable X does not occur, and

2) the variable X occurs in some other equation,

then substitute in all other equations every occurrence of the variable X by the term t.

g) If there is an equation of the form $X = t$ such that X is a variable and t is a non-variable term which contains this variable, then exit with failure (*not unifiable, positive occurs-check*).

h) If no transformation can be applied any more, then exit with success ($unifiable$).

Execution (Deransart)

1. Start with a *current goal* which is the initial definite goal G and a *current substitution* which is the empty substitution.
2. If G is `true` then stop (*success*), otherwise
3. Choose a predication `A` in G (*predication-choice*)
4. If `A` is `true`, delete it, and proceed to step (2), otherwise
5. If no freshly renamed clause in P has a head which unifies with `A` then stop (*failure*), otherwise
6. Choose in P a freshly renamed clause `H :- B` whose head unifies with `A` by substitution σ which is the *MGU* of `H` and `A` (*clause-choice*), and
7. Replace in G the predication `A` by the body `B`, flatten and apply the substitution σ to obtain the new current goal, let the new current substitution be the current substitution composed with σ, and proceed to step (2).

Exemples « Famille ».

```
/* ingalls_parent(P,E) est vrai ssi P est un parent de l'enfant E dans la
famille Ingalls réduite autour de Laura. */
ingalls_parent(lansford,peter).
ingalls_parent(lansford,charles).
ingalls_parent(charles,mary).
ingalls_parent(mary,adam).
ingalls_parent(charles,laura).
ingalls_parent(laura,rose).
ingalls_parent(charles,carrie).

/* ingalls_frere(F,P) est vrai ssi F est un frère (ou une soeur) de la personne
P pour la famille Ingalls réduite autour de Laura. */
ingalls_frere(X,Y):-ingalls_parent(Z,X), ingalls_parent(Z,Y), dif(X,Y).

/* ingalls_grandParent(G,E) est vrai ssi G est un grand-parent de l'enfant E
pour la famille Ingalls réduite autour de Laura. */
ingalls_grandParent(X,Y):-ingalls_parent(X,Z), ingalls_parent(Z,Y).

/* ingalls_ancetre(A,E) est vrai ssi A est un ancêtre de l'enfant E pour la
famille Ingalls réduite autour de Laura. */
ingalls_ancetre(X,A):- ingalls_parent(X,A).
ingalls_ancetre(X,A):- ingalls_parent(X,I), ingalls_ancetre(I,A).

/* Résultats d'exécutions
?- ingalls_parent(charles,laura).
true

?- ingalls_parent(X,laura).
X = charles;

?- ingalls_parent(charles,X).
X = mary;
X = laura;
X = carrie;

?- ingalls_grandParent(X,mary).
X = lansford; */
```

Exemples « Listes »

```
/* ajouteEnDernier(R,L,E) est vrai ssi R est obtenu à partir de L en ajoutant E
en dernier. */
ajouteEnDernier(E,[],[E]).
ajouteEnDernier(E,[F|L],[F|M]):-ajouteEnDernier(E,L,M).

/* concatene(D,F,L) est vrai si et seulement la concaténation des listes D et F
donne la liste L. */
concatene([],L,L).
concatene([E|L],M,[E|R]):-concatene(L,M,R).

/* supprimeDoublonConsecutif(L,R) est vrai ssi R est défini à partir de L en
supprimant les doublons consécutifs de L. */
supprimeDoublonConsecutif([E],[E]).
supprimeDoublonConsecutif([E,E|L],R):-supprimeDoublonConsecutif([E|L],R).
supprimeDoublonConsecutif([E,F|L],[E|R]):- dif(E,F),
  supprimeDoublonConsecutif([F|L],R).
```

Bibliographie.

Blackburn P., et al.., Prolog, tout de suite !, www.learnprolognow.org, Cahiers de Logique et d'Epistémologie, College Publications, 2007

Colmerauer, A. et al. *Le manuel de Prolog IV*, PrologIA, Marseille, alain.colmerauer.free.fr/alcol/ArchivesPublications/Prolog4Manuel.pdf, 1996.

Delahaye J.-P.,Cours de ProLog avec Turbo ProLog, Eyrolles, 1988.

Deransart P.et al.., Prolog : the standard, Springer, 1996.