Logique

- p :- a, b. ⇔ p = (a ∧ b) ∨ c.
  p :- c
- p :- a, !, b. ⇔ p = (a ∧ b) ∨ (¬a ∧ c)
  p :- c. Si ... ALORS ... SINON
- p :- c
  p :- a, !, b ⇔ c ∨ (a ∧ b).
- p(a) :- !. ⇒ X = a.
- Ex : (1) p(a).
  (2) p(X) :- q(X), r(X)
  (4) q(a)
  (3) p(X) :- u(X).
  (5) q(b).
  (6) q(c).
  (7) r(a).
  (8) r(b).
  (9) r(d).
  (10) u(d).

(2) → !, q(X), r(X) ⇒ X = a, X = a, X = b.
(2) → q(X), !, r(X) ⇒ X = a, X = a.

- minimum(X, Y, X) :- X =< Y, !. ⇒ accélère le prgm.
  minimum(X, Y, X) :- X > Y
- minimum(X, Y, X) :- X =< Y, ! COUPURE VERTE
  minimum(X, Y, Y). ⇒ rép. pas tjs correcte
- Cut rouge : ⇒ modifie les résultats.
  intersection([], _, []) :- !. ⇒ obligatoire !
  intersection([H|T1], L, [H|T2]) :- membre(H, L), !, intersection(T, L, T2).
  intersection([_|T1], L, T2) :- intersection(T, L, T2).
- Arrêt après la 1ère solution :
  premiereSolution(But) :- call(But), !.
- Négation :
  - diff(X, X) :- !, fail.
    diff(X, Y).
  - non(But) :- call(But), !, fail.
    non(But).

  ⇒ possible pour des variables non instanciées (∉ NON)

Prédicats ensemblistes

- setof. ?- setof(X, nom(X), S).
  ⇒ tous les noms.
  ?- setof(X, boisson(X, V), L).
  ⇒ pour chaque X affiche Y.
  ?- setof(X, boisson(X, lait), L).
- bagof. Idem mais retourne les doublons.
- findall ⇒ position fixée.
  ?- findall(X, boisson(X, Y), S). ⇒ 1ère pos.

Base de données

- op(priorité, Association, Nom).
  :- op(1100, fx, afficheListe) fx : a - b - c
  afficheListe(X, Y) :- nl,
  for(Y, write(X), nl),
  fail.
  ↖ backtracking
- bddVirt(NomBDD, Clé, nom du chp, Var. ass)

Ex : ?- afficheEltListe[(Prop, Marque, Couleur),
ce qu'on cherche
(bddVirt(Voiture, NI, Propriétaire, prop),
bddVirt(" " , Marque, mark),
bddVirt(" " , Couleur, color),
bddVirt(personne(nom, Prop, age, Age),
Age > 41))].

- Comment utiliser une BDD pour faire des calculs ?

Ex : qAge :- asserta(age(0,0)), for(personne(_, _, Age, _, _), qAge(Age), age(NbP, CumulAge), retract(age(X, Y))),
Moyenne is CumulAge/NbP,
write('Moyenne = '), write(Moyenne), nl.
CumulAge :- age(NbP, CumulAge),
retract(age(X, Y)),
NbPN is NbP + 1,
CumulAgeN is CumulAge + Age,
asserta(age(NbPN, CumulAgeN)).

Librairies

- x − y ≤ 3
  x + y ≤ 1
  x − 2 ≥ 3
  x + 2 ≥ 1
  −20 ≤ y, z ≤ 20.

:- use_module(library(clpr)).
f(X, Y, Z) :- {X − Y ≤ 3, X + Y ≤ 1,
−X + 2 ≤ −3, −X − 2 ≤ −1,
X ≥ −20, X ≤ 20, Y ≥ −20,
Y ≤ 20, Z = <20, Z >= −20,
maximize(Z),
maximize(X), maximize(Y).}

- use_module(library('clp/bounds')).

couleur(1, bleu).
couleur(2, jaune).
couleur(3, rouge).
colorier :- ListeVar = [C1, C2, C3, C4], // Déterminer les variables et les stocker dans une liste
ListeVar in 1...3, // Domaine de variation
label(ListeVar), // Lancement de l'algorithme
all_different([C1, C2]), // Contraintes
[...], // Mes ≠
afficheListe(ListeVar).

- chr

:- module(nbPremiers, premiers/1, premier/1).
celui qu'on veut — premiers du premier
:- use_module(library(chr)).
:- chr_constraint premiers/1, premier/1.
// opérateurs sur lesquels portent les contraintes
premiers(1) <=> true.
premiers(N) <=> N > 1 | M is N − 1, premier(N), premiers(M).
elimination(J) @ premier(I) \ premier(J) <=>
J mod I =:= 0 | true.