% Encodage en UTF-8

:- encoding(utf8).

% Prédicats d'affichage fournis

% set\_echo: ce prédicat active l'affichage par le prédicat echo

set\_echo :- assert(echo\_on).

% clr\_echo: ce prédicat inhibe l'affichage par le prédicat echo

clr\_echo :- retractall(echo\_on).

% echo(T): si le flag echo\_on est positionné, echo(T) affiche le terme T

% sinon, echo(T) réussit simplement en ne faisant rien.

echo(T) :- echo\_on, !, write(T).

echo(\_).

% ---------------------------------------------------------------------------

% Definition de l'operateur fourni

:- op(20, xfy, ?=).

% ---------------------------------------------------------------------------

% Partie unification

% Definition de occur\_check

occur\_check(X, T) :- var(X), contains\_var(X, T).

% Le var(X) est peut-etre inutile dans le cas présent car lorsqu'on appelle occur\_check, on a déjà testé si X est une variable.

% On le garde pour une définition plus rigoureuse, vu que peu couteux.

% Definition de regle(E, R)

% Definition des regles de transformation

% Definition de Rename

regle(X ?= T, rename) :- var(X), var(T).

% Definition de Simplify

regle(X ?= T, simplify) :- var(X), atom(T).

% Definition de Expand

regle(X ?= T, expand) :- var(X), compound(T), \+(occur\_check(X, T)).

% Definition de Check

regle(X ?= T, check) :- var(X), X \== T, occur\_check(X, T).

% Definition de Orient

regle(T ?= X, orient) :- var(X), nonvar(T).

% Definition de Decompose

regle(F ?= G, decompose) :- nonvar(F), nonvar(G),

functor(F, Ff, Fn), functor(G, Gf, Gn),

Ff == Gf, Fn =:= Gn.

% Definition de Clash

regle(F ?= G, clash) :- nonvar(F); nonvar(G),

functor(F, Ff, Fn), functor(G, Gf, Gn),

(Ff \== Gf; Fn =\= Gn), !.

% Definition du predicat d'unification de E avec la liste passée en paramètre,

suppr(E, [T|Q], S) :- E \== T, append([T], V, S), suppr(E, Q, V), !.

suppr(E, [T|Q], S) :- E == T, suppr(E, Q, S).

suppr(\_, [], []).

% Definition du predicat de suppression de la tete de liste

suppr\_T([\_|Q], Q).

suppr\_T([], []).

% Definition du predicat d'unification de deux listes.

% On unifie la tete de L1 avec celle de L2, puis on fait un appel récursif.

unif\_list([T1|Q1], [T2|Q2], L) :- append([T1 ?= T2], V, L), unif\_list(Q1, Q2, V).

unif\_list([], [], []).

% Definition de reduit(R,E,P,S) (avec S = Q de l'enonce)

reduit(rename, X ?= T, P, S) :- suppr(X ?= T, P, S), X = T.

reduit(simplify, X ?= T, P, S) :- suppr(X ?= T, P, S), X = T.

reduit(expand, X ?= T, P, S) :- suppr(X ?= T, P, S), X = T.

reduit(check, \_, \_, []) :- false.

reduit(orient, T ?= X, P, S) :- suppr(T ?= X, P, S2), append([X ?= T], S2, S).

reduit(decompose, F ?= G, P, S) :- F =.. Fliste, G =.. Gliste,

suppr\_T(Fliste, Freste),

suppr\_T(Gliste, Greste),

unif\_list(Freste, Greste, L),

suppr(F ?= G, P, V),

append(L, V, S).

reduit(clash, \_, \_, []) :- false.

% ---------------------------------------------------------------------------

% Definition des strategies

% Definition du predicat unifie

unifier([], \_).

unifier(P, Strategie) :- choix(Strategie, P, E, R),

afficher\_trace(R, E, P),

reduit(R, E, P, Q),

unifier(Q, Strategie), !.

% Definition du predicat de choix de strategie

choix(premier, P, E, R) :- choix\_premier(P, E, R).

choix(pondere, P, E, R) :- choix\_pondere(P, E, R).

% Definition de la liste ordonnee des regles representant l'echelle de poids

echelle\_poids([check, rename, simplify, orient, decompose, expand, clash]).

% echelle\_poids([expand, decompose, orient, simplify, rename, check, clash]).

% Definition du predicat trouver\_equation qui se charge de trouver une equation sur laquelle on peut appliquer une regle, et d'y appliquer la regle correspondante

trouver\_equation([R|\_], [E|\_], E, R) :- regle(E, R).

trouver\_equation(X, [\_|Q], E, R) :- trouver\_equation(X, Q, E, R).

trouver\_equation(\_, [], \_, \_) :- false.

% Definition du predicat selectionner\_regle qui se charge de trouver une regle applicable

selectionner\_regle(X, P, E, R) :- trouver\_equation(X, P, E, R).

selectionner\_regle([\_|Q], P, E, R) :- selectionner\_regle(Q, P, E, R).

% Definition de choix\_premier et de choix\_pondere.

% Choix d'une equation E grâce et de la regle R correspondante

choix\_premier([E|\_], E, R) :- regle(E, R), !.

choix\_pondere(P, E, R) :- echelle\_poids(X), selectionner\_regle(X, P, E, R), !.

% ---------------------------------------------------------------------------

% Predicats d'affichage

% menu : ce predicat va set\_echo, puis echo un menu d'accueil

menu :-

set\_echo, nl,

ansi\_format([fg(blue)], '===============================================', []), nl,

ansi\_format([fg(red)], ' MARTELLI - MONTANARI', []), nl,

ansi\_format([fg(red)], ' UNIFICATION PROGRAM', []), nl, nl,

ansi\_format([fg(green)], 'Utilisation du programme : unifie(X)', []), nl,

ansi\_format([fg(green)], '(avec X liste d\'équations de la forme (F ?= G)', []), nl,

ansi\_format([fg(blue)], '===============================================', []), nl, nl.

% afficher trace : affiche l'etat courant du programme

afficher\_trace(R, E, S) :- echo('System : '), echo(S), nl,

echo(R), echo(' : '), echo(E), nl.

% Predicats pour lancer l'unification avec ou non l'affichage de la trace

trace\_unif(P, Strategie) :- set\_echo, unifier(P, Strategie).

unif(P, Strategie) :- clr\_echo, unifier(P, Strategie).

% Appel des predicats pour l'affichage ou non de la trace en fonction du choix utilisateur

unifier\_trace(P, Strategie, oui) :- trace\_unif(P, Strategie).

unifier\_trace(P, Strategie, \_) :- unif(P, Strategie).

% affichage et choix de la strategie a appliquer

% strategie premier par defaut

unifie(P) :- echo('Choisissez une strategie : '), nl,

echo('(a) Premier'), nl,

echo('(b) Pondere'), nl,

read(NumStrat),

(NumStrat == a -> Strategie = premier ;

NumStrat == b -> Strategie = pondere;

Strategie = premier),

echo('Strategie choisie : '), echo(Strategie), nl,

echo('Activer la trace d\'affichage des règles ? (oui/non)'), nl,

read(Activer), nl,

echo('Trace activee : '), echo(Activer), nl,

unifier\_trace(P, Strategie, Activer).

% ---------------------------------------------------------------------------

% init : sera éxécuté dès le lancement du programme

init :- menu.

:- init.