Mahr Benjamin
Ginestra Alex
Logique et Modèle de Calculs

Projet LMC: code et tests

Dans ce rapport, nous fournissons le code du programme en prologue ainsi que les tests qui ont été effectué sur ce dernier.

Code:

```
:- op(20,xfy,?=).
% 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(_).

% Question 1
% Predicat occur_check : return true if a variable V is contained in T.
% T n'est pas une fonction (T est soit une variable soit une constante):
occur_check(V,T) :- V == T, !.

% T est un fonction:
occur_check(V,T) :- compound(T),functor(T,_,A),occur_check_base(V,T,A).

% T est un fonction avec une arité de 1:
occur_check_base(V,T,1) :- arg(1,T,X),!,occur_check(V,X).

% T est un fonction avec un arité supérieur à 1:
occur_check_base(V,T,N) :- arg(N,T,X),occur_check(V,X);N1 is
(N-1),occur_check_base(V,T,N1).

% Predicat regle: il verifie qu'une regle passée en deuxieme parametre peut s'appliquer sur une
expression passée en premier paramètre.

% Listes des predicats regles:
regle((_ ?= T),rename) :- var(T),!.

regle((_ ?= T),simplify) :- atomic(T),!.
```

```
regle((X ?= T),check) :- X \== T, var(X), occur_check(X,T),!.

regle((X ?= T),expand) :- var(X), compound(T), not(occur_check(X,T)), !.

regle((X ?= T),decompose) :- compound(X), compound(T), functor(X,A1,N1),
functor(T,A2,N2),A1==A2,N1==N2,!.

regle((X ?= T),clash) :- compound(X), compound(T), functor(X,A1,_), functor(T,A2,_), A1 \==
A2, !.

regle((X ?= T),clash) :- compound(X), compound(T), functor(X,_,N1), functor(T,_,N2), N1 \==
N2, !.

regle((T ?= _),orient) :- nonvar(T),!.

% prédicat qui vont servir pour creer les prédicats réduits
append(X,[],X).
append(Y,[X|P],[X|Q]) :- append(Y,P,Q).

decomposer((X ?= T),1,L1,[A ?= B|L1]) :- arg(1,X,A),arg(1,T,B),!.

decomposer((X ?= T),N,L1,L2) :- N2 is (N-1),arg(N,X,A),arg(N,T,B),decomposer(X ?= T,N2,[A ?
= B|L1],L2).

% ajout du prédicat substitution pour les prédicats réduits
% on remplace premier parametre par deuxieme parametre dans équation du troisieme
parametre
% et on met le resultat dans le quatrieme parametre
substitution(_,_,[],[]) :- !.

substitution(X,T,[A ?= B|P],[A2 ?= B2|P2]) :-
substitution_terme(X,T,A,A2),substitution_terme(X,T,B,B2),substitution(X,T,P,P2).

substitution_terme(X,T,A,T):- A == X,not(compound(A)),!.

substitution_terme(X,_,A,A):- A \== X,not(compound(A)),!.

substitution_terme(X,T,A,Q):- functor(A,_,N),compound(A),substitution_funct(X,T,A,N,Q),!.

% Liste predicats pour subsitution dans une fonction
substitution_funct(X,T,A,1,Q):-
functor(A,F,N),arg(1,A,B),substitution_terme(X,T,B,V),functor(Q,F,N),arg(1,Q,V),!.

substitution_funct(X,T,A,N,Q):-
functor(A,F,M),arg(N,A,B),substitution_terme(X,T,B,V),functor(Q,F,M),arg(N,Q,V),N2 is
(N-1),substitution_funct(X,T,A,N2,Q),!.

% Predicat suivant creer pour ne pas faire de boucle
substitution_autre(_,_,[],[]):- !.

substitution_autre(X,T,[A=B|P],[A2=B2|P2]):-
substitution_terme(X,T,A,A2),substitution_terme(X,T,B,B2),substitution_autre(X,T,P,P2).
```

```
% fin predicats pour reduit, donc predicats reduit
% Liste des prédicats réduits:
reduit(decompose,(X ?= Y),P1;Q,P2;Q):- echo(system :[X = Y|P1]),echo('\n'),echo(decompose :
(X = Y)),echo('\n'),functor(X,_,A),decomposer(X ?= Y,A,[],L),append(P1,L,P2),!.

reduit(rename,(X ?= Y),P1;Q1,P2;[X=Y|Q2]):- echo(system :[X = Y|
P1]),echo('\n'),echo(rename :(X =
Y)),echo('\n'),substitution(X,Y,P1,P2),substitution_autre(X,Y,Q1,Q2),!.

reduit(simplify,(X ?= Y),P1;Q1,P2;[X=Y|Q2]):- echo(system :[X = Y|
P1]),echo('\n'),echo(simplify :(X =
Y)),echo('\n'),substitution(X,Y,P1,P2),substitution_autre(X,Y,Q1,Q2),!.

reduit(expand,(X ?= Y),P1;Q1,P2;[X=Y|Q2]):- echo(system :[X = Y|
P1]),echo('\n'),echo(expand :(X =
Y)),echo('\n'),substitution(X,Y,P1,P2),substitution_autre(X,Y,Q1,Q2),!.

reduit(orient,(X ?= Y),P;Q,[X ?= Y|P];Q):- echo(system :[X = Y|P]),echo('\n'),echo(orient :(X =
Y)),echo('\n'),!.

reduit(check,(X ?= Y),P;Q,P;Q):- echo(system :[X = Y|P]),echo('\n'),echo(check :(X =
Y)),echo('\n'),write('\n No'),fail,!.

reduit(clash,(X ?= Y),P;Q,P;Q):- echo(system :[X = Y|P]),echo('\n'),echo(clash :(X =
Y)),echo('\n'),write('\n No'),fail,!.


% Question 2

% Echelle de poids indiqué :
poids(clash,5).
poids(check,5).
poids(rename,4).
poids(simplify,4).
poids(orient,3).
poids(decompose,2).
poids(expand,1).

% predicats pour choix pondere et choix_premier
ordrePoids([X],R,X):- regle(X,R),!.
ordrePoids([X,T|P],R,E):- regle(X,R1),poids(R1,P1),regle(T,R2),poids(R2,P2),((P1 >= P2)-
>ordrePoids([X|P],R,E);ordrePoids([T|P],R,E)).

% pour afficher à les variables de depart
println([]):- !.
println([X=T|P]):- echo(X = T),echo('\n'),println(P).

% predicats des choix d application
choix_premier([],Q,_,_):- echo('\n'),println(Q),echo('\n'),write('Yes'),!.
choix_premier([E|P],Q,E,R):- regle(E,R),reduit(R,E,P;Q,P2;Q2),choix_premier(P2,Q2,_,_).
```

```
retirerElem(_,[],[]):- !.
retirerElem(X,[T|R],V):- ((X == T)->(V = R),!; retirerElem(X,R,V)).

choix_pondere([],Q,_,_):- echo('\n'),println(Q),echo('\n'),write('Yes'),!.
choix_pondere(P1,Q,E,R):-
ordrePoids(P1,R,E),retirerElem(E,P1,P2),reduit(R,E,P2;Q,P3;Q3),choix_pondere(P3,Q3,_,_).

unifie(P,premier):- choix_premier(P,_,_,_).
unifie(P,pondere):- choix_pondere(P,_,_,_).
unifie(P):- choix_premier(P,_,_,_).

% Question 3

trace_unif(P,Strategie):- set_echo,unifie(P,Strategie),clr_echo,!.
unif(P,Strategie):- clr_echo,unifie(P,Strategie),clr_echo, !.

% amelioration affichage
instruction(P,Strategie,oui) :- trace_unif(P,Strategie), !.
instruction(P,Strategie,non) :- unif(P,Strategie), !.

option(oui):- demarrer.
option(non):- fail, !.

fin:- write('\n \t Souhaitez-vous continuer a utiliser le programme? Entrer oui ou non \n'),write('
\t Choix:'),read(Choix),option(Choix), !.

demarrer:- write('\t Ce programme repose sur l\'Algorithme d\'unification de Martelli-
Montanari \n \n Nb: N\'oubliez pas d\'ajouter un point a chaque entre \n'),write('\n Ecrire le
systeme que vous souhaitez unifier: \n'), write(' \t Systeme equation:'), read(Sys),write('\n
Differents choix de strategie: laquelle desirez-vous utiliser ? Entrez premier ou pondere
\n'),write('\t Strategie:'),read(Strat),write('\n Souhaitez-vous faire apparaitre dans le terminal la
trace ? Entrer oui ou non \n'),write('\t
Choix:'),read(Choix),write('\n'),instruction(Sys,Strat,Choix),fin, !.
```

Tests:

Premier test: [f(X,Y)?=f(g(Z),h(a)),Z?=f(Y)].

Choix_premier:

Ce programme repose sur l'Algorithme d'unification de Martelli-Montanari
Nb: N'oubliez pas d'ajouter un point a chaque entre
Ecrire le système que vous souhaitez unifier:
Système equation:[f(X,Y)?=f(g(Z),h(a)),Z?=f(Y)].
Différents choix de stratégie: laquelle désirez-vous utiliser ? Entrez premier ou pondère
Strategie:|: premier.
Souhaitez-vous faire apparaitre dans le terminal la trace ? Entrer oui ou non
Choix:|: oui.

```
system:[f(_2316,_2318)=f(g(_2322),h(a)),_2322?=f(_2318)]
decompose:(f(_2316,_2318)=f(g(_2322),h(a)))
```

system:[_2316=g(_2322),_2318?=h(a),_2322?=f(_2318)]
expand:(_2316=g(_2322))
system:[_2318=h(a),_2322?=f(_2318)]
expand:(_2318=h(a))
system:[_2322=f(h(a))]
expand:(_2322=f(h(a)))

_2322=f(h(a))
_2318=h(a)
_2316=g(f(h(a)))

Yes

Choix_pondere:
Ecrire le système que vous souhaitez unifier:
Systeme equation:|: [f(X,Y)?=f(g(Z),h(a)),Z?=f(Y)].
Différents choix de stratégie: laquelle désirez-vous utiliser ? Entrez premier ou pondère
Strategie:|: pondere.
Souhaitez-vous faire apparaitre dans le terminal la trace ? Entrer oui ou non
Choix:|: oui.

system:[f(_2886,_2888)=f(g(_2892),h(a)),_2892?=f(_2888)]
decompose:(f(_2886,_2888)=f(g(_2892),h(a)))
system:[_2886=g(_2892),_2888?=h(a),_2892?=f(_2888)]
expand:(_2886=g(_2892))
system:[_2888=h(a),_2892?=f(_2888)]
expand:(_2888=h(a))
system:[_2892=f(h(a))]
expand:(_2892=f(h(a)))

_2892=f(h(a))
_2888=h(a)
_2886=g(f(h(a)))

Yes

Deuxième test:[f(X,Y)?= f(g(Z),h(a)),Z?=f(X)].

Choix_premier: Ecrire le systeme que vous souhaitez unifier:
Systeme equation:[f(X,Y)?= f(g(Z),h(a)),Z?=f(X)].

Differents choix de strategie: laquelle desirez-vous utiliser ? Entrez premier ou pondere
Strategie:|: premier.

Souhaitez-vous faire apparaitre dans le terminal la trace ? Entrer oui ou non
Choix:|: oui.

system:[f(_598,_600)=f(g(_604),h(a)),_604?=f(_598)]
decompose:(f(_598,_600)=f(g(_604),h(a)))
system:[_598=g(_604),_600?=h(a),_604?=f(_598)]
expand:(_598=g(_604))
system:[_600=h(a),_604?=f(g(_604))]
expand:(_600=h(a))
system:[_604=f(g(_604))]
check:(_604=f(g(_604)))

No

Choix pondéré: Ecrire le systeme que vous souhaitez unifier:
Systeme equation:[f(X,Y)?= f(g(Z),h(a)),Z?=f(X)].

Differents choix de strategie: laquelle desirez-vous utiliser ? Entrez premier ou pondere
Strategie:|: pondere.

Souhaitez-vous faire apparaitre dans le terminal la trace ? Entrer oui ou non
Choix:|: oui.

system:[f(_580,_582)=f(g(_586),h(a)),_586?=f(_580)]
decompose:(f(_580,_582)=f(g(_586),h(a)))
system:[_580=g(_586),_582?=h(a),_586?=f(_580)]
expand:(_580=g(_586))
system:[_586=f(g(_586))]
check:(_586=f(g(_586)))

No

Troisième test:[f(X,Y)?= f(g(Z),h(a)),Z?=f(R),R?=f(Y)].

Choix_premier:Ecrire le systeme que vous souhaitez unifier:
Systeme equation:[f(X,Y)?= f(g(Z),h(a)),Z?=f(R),R?=f(Y)].

Differents choix de strategie: laquelle desirez-vous utiliser ? Entrez premier ou pondere
Strategie:|: premier.

Souhaitez-vous faire apparaitre dans le terminal la trace ? Entrer oui ou non
Choix:|: oui.

system:[f(_580,_582)=f(g(_586),h(a)),_586?=f(_612),_612?=f(_582)]
decompose:(f(_580,_582)=f(g(_586),h(a)))
system:[_580=g(_586),_582?=h(a),_586?=f(_612),_612?=f(_582)]
expand:(_580=g(_586))
system:[_582=h(a),_586?=f(_612),_612?=f(_582)]
expand:(_582=h(a))
system:[_586=f(_612),_612?=f(h(a))]

expand:(_586=f(_612))
system:[_612=f(h(a))]
expand:(_612=f(h(a)))

_612=f(h(a))
_586=f(f(h(a)))
_582=h(a)
_580=g(f(f(h(a))))

Yes

Choix pondéré: Ecrire le systeme que vous souhaitez unifier:
Systeme equation:|: [f(X,Y)?= f(g(Z),h(a)),Z?=f(R),R?=f(Y)].

Differents choix de strategie: laquelle desirez-vous utiliser ? Entrez premier ou pondere
Strategie:|: pondere.

Souhaitez-vous faire apparaitre dans le terminal la trace ? Entrer oui ou non
Choix:|: oui.

system:[f(_1462,_1464)=f(g(_1468),h(a)),_1468?=f(_1494),_1494?=f(_1464)]
decompose:(f(_1462,_1464)=f(g(_1468),h(a)))
system:[_1462=g(_1468),_1464?=h(a),_1468?=f(_1494),_1494?=f(_1464)]
expand:(_1462=g(_1468))
system:[_1464=h(a),_1468?=f(_1494),_1494?=f(_1464)]
expand:(_1464=h(a))
system:[_1468=f(_1494),_1494?=f(h(a))]
expand:(_1468=f(_1494))
system:[_1494=f(h(a))]
expand:(_1494=f(h(a)))

_1494=f(h(a))
_1468=f(f(h(a)))
_1464=h(a)
_1462=g(f(f(h(a))))

Yes

FIN TEST

Sources:
https://perso.liris.cnrs.fr/christine.solnon/prolog.html
https://perso.liris.cnrs.fr/nathalie.guin/Prolog/Cours/Cours2-Prolog1.pdf
http://www.swi-prolog.org (différentes rubriques)