|  |
| --- |
| Projet LMC  Algorithme d’unification de Martelli-Montanari en Prolog |
| V DAMM – F. NOSARI - 6-déc.-16 |
| Table des matières  [Règles d’unification 3](#_Toc468830177)  [Teste d’applicabilité des règles 4](#_Toc468830178)  [Rename 4](#_Toc468830179)  [Simplify 4](#_Toc468830180)  [Expand 4](#_Toc468830181)  [Check 5](#_Toc468830182)  [Orient 5](#_Toc468830183)  [Decompose 5](#_Toc468830184)  [Clash 5](#_Toc468830185)  [Application des règles 7](#_Toc468830186)  [Rename 7](#_Toc468830187)  [Simplify 7](#_Toc468830188)  [Expand 8](#_Toc468830189)  [Check 8](#_Toc468830190)  [Orient 8](#_Toc468830191)  [Decompose 8](#_Toc468830192)  [Clash 9](#_Toc468830193)  [Unification 10](#_Toc468830194) |

# Règles d’unification

La liste des règles applicables est la suivante :

1. Rename {x ?= t} ∪ P ′ ; S → P ′ [x/t]; S[x/t] ∪ {x = t} si t est une variable
2. Simplify {x ?= t} ∪ P ′ ; S → P ′ [x/t]; S[x/t] ∪ {x = t} si t est une constante
3. Expand {x ?= t} ∪ P ′ ; S → P ′ [x/t]; S[x/t] ∪ {x = t} si t est composé et x n’apparaît pas dans t
4. Check {x ?= t} ∪ P ′ ; S → ⊥ si x 6= t et x apparaît dans t
5. Orient {t ?= x} ∪ P ′ ; S → {x ?= t} ∪ P ′ ; S si t n’est pas une variable
6. Decompose {f(s1, · · · , sn) ?= f(t1, · · · , tn)} ∪ P ′ ; S → {s1 ?= t1, · · · , sn ?= tn} ∪ P ′ ; S
7. Clash {f(s1, · · · , sn) ?= g(t1, · · · , tm)} ∪ P ′ ; S → ⊥ si f 6= g ou m 6= n

## Teste d’applicabilité des règles

Pour chaque règle on créer un prédicat regle(E, r) tel que le prédicat est vrai si on peut appliquer la règle r à l’équation E.

### Rename

Pour la règle rename on doit vérifier si t est une variable et si x en est une également.

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

### Simplify

Pour la règle simplify on doit vérifier si t et une constante et si x est une variable.

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

### Expand

Pour la règle expand on doit vérifier si t et composé, si x est une variable et si x n’apparait pas dans t.

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

Le prédicat occur\_check(X,T) est décrit dans la section « Prédicats annexes ».

### Check

Pour la règle check on doit vérifier si x est une variable et si x apparait dans t.

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

Le prédicat occur\_check(X,T) est décrit dans la section « Prédicats annexes », il est vrai si X apparait dans T.

### Orient

Pour la règle orient on doit vérifier si x est une variable et si t n’est pas une variable.

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

### Decompose

Pour la règle decompose on doit vérifier si f(s1, ···, sn) et f(t1, ···, tn) sont deux fonctions de même symbole et de même arité.

regle(S?=T,decompose) :- compound(S), compound(T),!,

compound\_name\_arity(S,SF,SA),

compound\_name\_arity(T,TF,TA), SF==TF, SA==TA.

### Clash

Pour la règle clash on doit vérifier si f(s1, ···, sn) et g(t1, ···, tn) sont deux fonctions de même symbole ou de même arité.

regle(F?=G,clash) :- **compound**(F), **compound**(G),!,

compound\_name\_arity(F,FF,FA),

compound\_name\_arity(G,GF,GA),

not( (FF == GF, FA == GA) ).

## Application des règles

Pour l’application d’une règle r on créer le predicat reduit(r,E,P,Q) qui transforme le système d’équations P en le système d’équations Q par application de la règle r de transformation sur l’équation E.

Par soucis de compréhension et pour une meilleur lisibilité du code nous avons décomposé l’application d’une règle en 2 prédicats tel que :

r(E,P,R) ← ….

reduit(r,E,P,Q) ← r(E,P,Q).

### Rename

La règle rename remplace x en t (x=t) et propage cette modification dans le reste du système d’équation.

rename(X?=T,[X?=T|L],Q) :- X=T,Q=L.

reduit(rename,E,P,Q) :- rename(E,P,Q).

### Simplify

La règle simplify remplace x en t (x=t) et propage cette modification dans le reste du système d’équation.

simplify(X?=T,[X?=T|L],Q) :- X=T,Q=L.

reduit(simplify,E,P,Q) :- simplify(E,P,Q).

### Expand

La règle expand remplace x en t (x=t) et propage cette modification dans le reste du système d’équation.

expand(X?=T,[X?=T|L],Q) :- X=T,Q=L.

reduit(expand,E,P,Q) :- expand (E,P,Q).

### Check

La règle check ne fait rien d’autre que d’échouer, en effet si elle est applicable alors l’unification n’est pas possible.

Puisque qu’elle échoue toujours on peut la remplacer par le prédicat fail.

### Orient

La règle orient remplace inverse t et x pour obtenir x?=t.

orient(T?=X,[T?=X|L],Q) :- **append**([X?=T],L,Q).

reduit(orient,E,P,Q) :- orient(E,P,Q).

### Decompose

La règle decompose remplace f(s1, ···, sn)?=f(t1, ···, tn) par s1?=t1, ···, sn?=tn dans le système d’équation.

decompose(X?=T,[X?=T|L],Q) :- term\_params(X, XL), term\_params(T, TL),

make\_list(XL, TL, LR), **append**(LR,L,Q).

reduit(decompose,E,P,Q) :- decompose(E,P,Q).

Le prédicat term\_param(T,L) est décrit dans la section « Prédicats annexes », il attribue à L les paramètres de T.

### Clash

La règle clash ne fait rien d’autre que d’échouer, en effet si elle est applicable alors l’unification n’est pas possible.

Puisque qu’elle échoue toujours on peut la remplacer par le prédicat fail.

# Unification

L’unification consiste à appliquer une règle si on le peut. On utilise donc le prédicat regle(E, r) pour tester si la règle r est applicable et si tel est le cas alors on utilise le prédicat reduit(r,E,P,Q) pour réduire le système d’équations.

Pour cela on créer le prédicat unifie(P) qui unifie le système d’équation P tel que :

unifie([X?=T|L]) ← regle(E,r), !, reduit(r,X?=T,[X?=T|L],Q), unifie(Q).

On teste si la règle est applicable, si on ne peut pas le prédicat échoue (cut), sinon on réduit le système d’équation jusqu’à avoir un système d’équation vide ou une unification impossible (aucune règle applicable).

Exemple pour la règle rename :

unifie([X?=T|L]) :- regle(X?=T,rename), reduit(rename,X?=T,[X?=T|L],Q),

unifie(Q), !.

unifie([]) :- echo('Fin de l\'unification\n').

Il y a cependant 2 exceptions, comme dit précédemment, les règles check et clash sont remplacées par le prédicat fail, elles sont donc faites de la façon suivante :

unifie([X?=T|L]) ← regle(E,r), !, fail.

Exemple pour la règle check :

unifie([X?=T|L]) :- regle(X?=T,rename), fail, !.

# Stratégies