# Algorithme d’unification

2018-06-10

# Table des matières

# 1 Prérequis

On décrira ici les types utilisés de l'algorithme sans rentrer dans les détails de l'implémentation.

## 1.1 Type Terme

### 1.1.1 Structure de données

```
TYPE Terme
```

On suit ici la définition du cours :

- Les **constantes** sont des termes;
- Les **variables** sont des termes;
- Un **prédicat n-aire** (d'arité n) s'applique sur un n-tuple de termes et produit une expression logique (un terme).
- Une **fonction n-aire** s'applique sur un n-tuple de termes et produit un terme.

Pour les curieux, on peux implémenter ce type en utilisant les types sommes. Il représente un type contenant toutes les valeurs de chacun des types A, B, … de telle sorte qu'une valeur issue de A ne puisse pas être confondue avec une valeur issue de B et ainsi de suite (même si A = B).

### 1.1.2 Fonctions utilisées

Renvoie vrai si le terme est un atome.

```
ALGO estUnAtome
ENTREES
  terme : Terme PAR VALEUR
SORTIES
  atome : BOOLEEN
```

Renvoie la liste des termes où s'applique la fonction ou le predicat

```
ALGO retournerArguments
ENTREES
  terme : Terme PAR VALEUR
SORTIES
  termes : LISTE de Terme
```

## 1.2 Type Couple

### 1.2.1 Structure de données

```
TYPE STRUCTURE Couple
  gauche : Terme
  droite : Terme
FINSTRUCTURE
```

### 1.2.2 Primitives

#### Constructeur

Construit un élement de type Couple

```
ALGO creerCouple
ENTREES
  elementGauche:Terme PAR VALEUR
  elementDroite : Terme PAR  VALEUR
SORTIES
  unCouple : Couple
```

#### Accesseurs

Retourne la valeur de l'élement **gauche** du couple.

```
ALGO gauche
ENTREES
  Couple : Couple PAR VALEUR
SORTIES
  termeGauche : Terme
```

Retourne la valeur de l'élement **droite** du couple.

```
SORTIES
ALGO droite
ENTREES
  Couple : Couple PAR VALEUR
SORTIES
  termeDroit : Terme
```

### Fonctions annexes

Renvoie vrai si les termes du couple sont égaux syntaxiquement.

```
ALGO egaliteSyntaxique
ENTREES
  Couple : Couple PAR VALEUR
SORTIES
  sontEgaux: BOOLEEN
```

Renvoie vrai si les termes du couple ont la même arité. Exception : Renvoie faux si le couple contient des variables ou des constantes.

```
ALGO sontDeMemeArite
ENTREES
  Couple : Couple PAR VALEUR
SORTIES
  memeArite: BOOLEEN
```

### 1.3 Type Liste de Substitution

*cf : cours instances et substitutions*

Renvoie vrai si un terme est déja substitué dans une liste de substitutions

```
ALGO subitUneSubstitution
ENTREES
  unTerme : Terme PAR VALEUR
  substitutions : Liste de Substitution PAR VALEUR
SORTIES
  modif: BOOLEEN
```

Renvoie vrai si un terme est une instance de l'application d'une substitution d'un terme

```
ALGO estUneInstance
ENTREES
  uneInstance : Terme PAR VALEUR
  unTerme : Terme PAR VALEUR
  substitutions : Liste de Substitution PAR VALEUR
SORTIES
  instance: BOOLEEN
```

Ajoute un substitution à la liste des substitutions

```
ALGO ajoutSubstitution
ENTREES
 termeASubstituer : Terme PAR VALEUR
  unTerme : Terme PAR VALEUR
  substitutions : Liste de Substitution PAR REFERENCE
SORTIES
 substitutions : : Liste de Substitution
```

Renvoie la substitution d'un terme

```
ALGO substitue
ENTREES
  termeASubstituer : Terme PAR VALEUR
  substitutions : Liste de Substitution PAR valeur
SORTIES
  uneSubstitution: Substitution
```

# 2 Algorithme

Il suit les conventions algorithmiques vu en Licence 1 sans introduire de nouveaux concepts et n'est pas factorisé intentionnellement.

## 2.1 Unification

```
ALGO Unification

ENTREES
  ensembleAUnifier : ENSEMBLE DE Couple PAR REFERENCE
  sigma : LISTE de Substitution PAR REFERENCE
  applicationRegle : BOOLEEN PAR VALEUR
SORTIES
  applicationRegle : BOOLEEN

VARIABLES
  unCouple : Couple
  i : NUMERIQUE
  argumentGauche : LISTE DE Terme
  argumentDroite : LISTE DE Terme
  termeGauche : Terme
  termeDroit : Terme
```

### 2.1.1 Initialisation

```
DEBUT
  argumentGauche <- []
  argumentDroite <- []

  unCouple <- choisir (ensembleAUnifier)
  termeGauche <- gauche(unCouple)
  termeDroit <- droite(UnCouple)
  applicationRegle <- FAUX
```

### 2.1.2 Suppression

```
SI egaliteSyntaxique(unCouple) ALORS

  supprimer(unCouple,ensembleAUnifier)

  // Unification terminée
  SI vide(ensembleAUnifier) ALORS
    RETOURNER VRAI , sigma

  SINON
    unCouple <- choisir (ensembleAUnifier)
    termeGauche <- gauche(unCouple)
    termeDroit <- droite(UnCouple)
    applicationRegle <- VRAI
  FINSI
FINSI
```

### 2.1.3 Décomposition

```
SI sontDeMemeArite(unCouple) ALORS

  argumentGauche<-retournerArguments (gauche(unCouple))
  argumentDroite<-retournerArguments (droite(unCouple))

  // ([a;b;c][d,e,f],{}) -> {(a,d);(b,e);(c,f)}
  POUR i ALLANT DE debut 1 A longueur(argumentGauche) FAIRE
    ajouter
      (
      creerCouple(argumentGauche[i] , argumentDroite[i]),
      ensembleAUnifier
      )
  FINPOUR

  supprimer(unCouple,ensembleAUnifier)
  unCouple <- choisir (ensembleAUnifier)
  termeGauche <- gauche(unCouple)
  termeDroit <- droite(UnCouple)
  applicationRegle <- VRAI
FINSI
```

### 2.1.4 Association

On applique cette conditionnelle dans l'autre sens également SI estUnAtome(termeDroit) ...

```
1   SI estUnAtome(termeGauche) ET
2      NON estUnAtome(termeDroit) ET
3      NON subitUneSubstitution(termeGauche , sigma) ET
4      NON estUneInstance(termeGauche,termeDroit , sigma) ALORS
5
6     ajoutSubstitution(termeGauche , termeDroit , sigma)
7     applicationRegle <- VRAI
8
9   FINSI
```

### 2.1.5 Fusion

On applique cette conditionnelle dans l'autre sens également SI estUnAtome(termeDroit) ...

```
1   SI estUnAtome(termeGauche) ET
2      NON egaliteSyntaxique(unCouple) ET
3      subitUneSubstitution(termeGauche , sigma) ALORS
4
5     supprimer(UnCouple , ensembleAUnifier)
6     ajouter (
7       creerCouple ( substitue(termeGauche) , termeDroit ),
8       ensembleAUnifier)
9     applicationRegle <- VRAI
10
11  FINSI
```

### 2.1.6 Récursion

```
1
2   SI NON applicationRegle OU vide(ensembleAUnifier) ALORS
3     RETOURNER applicationRegle , sigma
4   SINON
5     RETOURNER unification(ensembleAUnifier , sigma , applicationRegle)
6   FINSI
7
8 FIN
```

### 2.2 Lanceur

Hypothèse : Un ensemble vide de couple est unifiable

```
ALGO LanceurUnification
ENTREES
  ensembleAUnifier : ENSEMBLE DE Couple PAR VALEUR
SORTIES
  unifiable : BOOLEEN
  unificateur : LISTE DE Substitution
VARIABLES
  unificateur : LISTE DE Substitution
DEBUT

  // Pas de termes à unifier
  SI vide (ensembleAUnifier) ALORS
    RETOURNER VRAI , []

  // On applique l'algorithme
  SINON
    RETOURNER unification(ensembleAUnifier , [] , FAUX)

  FINSI

FIN
```