Cours 6 - Contrôle du Backtracking

Arsène Kabuya

ESIS Salama

L2 Génie Logiciel

Mai 2021

Contrôle du Backtracking

Introduction

Prolog effectue un backtracking (retour arrière) si nécessaire, dans le processus de satisfaction d'un but.

Le backtracking automatique est un atout puissant de Prolog. Toutefois, lorsqu'il n'est pas contrôlé, il peut rendre un programme inefficace.

Contrôle du Backtracking

La coupure

Nous pouvons contrôler ou empêcher le Backtracking en faisant usage de la **coupure**.

La coupure s'écrit dans le code Prolog avec le symbole « ! ».

Elle est utile car elle peut rendre nos programmes plus efficaces. En effet, à l'aide de la coupure, nous disons explicitement à Prolog de ne pas explorer d'autres alternatives, car nous savons qu'elles ne réussiront pas.

Contrôle du Bactracking

Exemple

Sans coupure

```
niveau(X,normal):- X<3.
niveau(X,alert1):- 3=<X, X<6.
niveau(X,alert2):- 6=<X.
```

Avec coupure

```
niveau2(X,normal):- X<3,!.
niveau2(X,alert1):- 3=<X, X<6,!.
niveau2(X,alert2):- 6=<X.
```

Contrôle du Backtracking

Fonctionnement de la coupure

Soit une clause de la forme :
T :- B1, B2, ..., Bm, !, Bn, ..., Br.

Supposons qu'une requête avec une clause **G** s'unifie avec la clause **T**. Pendant le parcours du corps de **T**, lorsque Prolog rencontre et exécute la coupure, la solution de **B1, ..., Bm** se fige, et toutes les possibles alternatives sont rejetées.

Toute tentative d'unifier la clause **G** avec une autre clause de tête **T** est aussi exclue. Toutefois, le backtracking reste possible pour les buts **Bn, ..., Br**, qui viennent après la coupure.

Contrôle du Backtracking

Exemples d'utilisation

La coupure permet aussi de ressortir le caractère très expressif de Prolog. On peut ainsi définir des règles mutuellement exclusives de la forme :
si **Condition** *alors* **Conclusion1**,
sinon **Conclusion2**.

Ex :

Sans la coupure

```
max(X,Y,X) :- X>=Y.

max(X,Y,Y) :- X<Y.
```

Avec la coupure

```
max2(X,Y,Max) :- X >=Y,!,
                 Max = X.
max2(X,Y,Y).
```

Contrôle du Backtracking

Exemples d'utilisation (suite)

Recherche toutes les occurrences de **X**

```
element(X,[X|L]).
element(X,[Y|L]) :- element(X,L).
```

Recherche une seule occurrence de `X`, la toute première :

```
element2(X,[X|L]):-!.
element2(X,[Y|L]) :- element2(X,L).
```

La négation par l'échec

Exprimer en Prolog :
« Luc aime tous les animaux sauf le renard. »

Reformulation :
Si X est un renard, alors il n'est pas vrai que « Luc aime X »,
sinon si X est un animal, alors Luc aime X.

Et en Prolog :

```
aime(luc,X) :- X = renard,!, fail.
aime(luc,X) :- animal(X).
```

De manière compacte, en une clause :

```
aime(luc,X) :- X = renard,!, fail ; animal(X).
```

La négation par l'échec

Exprimer en Prolog :
« X est différent de Y. »

Reformulation :
Si X s'unifie avec Y, alors `different(X,Y)` *échoue,*
sinon `different(X,Y)` *réussit.*

Et en prolog :

```
different(X,X) :- !, fail.
different(X,Y).
```

En une clause :

```
different(X,Y) :- X = Y,!, fail ; true.
```

La négation par l'échec

Le prédicat **fail** est un but qui échoue toujours.

Le prédicat **true** est un but qui réussit toujours.

La négation est définie en Prolog comme suit :

not(But) est vrai si **But** n'est pas vrai.

Nous pouvons le comprendre comme suit :

Si **But** *réussit alors* **not(But)** *échoue,*
sinon **not(But)** *réussit.*

La négation par l'échec

Et en Prolog :

```
not(P) :- P,!, fail ; true.
```

not(But) peut s'écrire aussi \+ **But**.

Cette négation est appelée *négation par l'échec* car elle ne correspond pas à la négation au sens mathématique.

Le prédicat **different(X,Y)** peut donc s'écrire :

```
different(X,Y) :- not(X = Y).
```

La négation par l'échec

Comportement de la négation

Dans la base de connaissance:

```
animal(loup).
```

Dans l'interpréteur :

```
?- animal(vache).
false.

?- not(animal(vache)).
true.
```

La négation par l'échec

Hypothèse du monde clos

La négation par l'échec est basée sur l'hypothèse du monde clos :

« Tout programme Prolog est supposé déclarer *tout* ce qui est vrai dans le monde décrit par le programme. Par conséquent, tout ce qui n'est pas déclaré par le programme (ou ne peut pas être déduit du programme) est supposé faux, et donc sa négation vraie. »

Le programme déclare **animal(loup)**, mais ne dit rien à propos de la vache. Prolog déduit que **animal(vache)** est faux, que **not(animal(vache))** devrait être vrai.

Les automates

Introduction

Un automate à états finis est un modèle d'un système et de son évolution, c'est-à-dire une description formelle du système et de la manière dont il se comporte.

Les automates finis sont l'un des modèles de calcul les plus anciens en informatique.

Actuellement, ils sont aussi utilisés dans l'analyse lexicale au niveau des compilateurs, dans le traitement des langues naturelles, dans le traitement de texte, dans les circuits et tables logiques, dans le processus de vérification des programmes, ...

Les automates

Définition

Un automate fini $\mathcal{A} = \langle E, \Sigma, \delta, e_0, F \rangle$ sur l'alphabet Σ est composé d'un ensemble fini d'états E, d'une fonction de transition $\delta : E \times \Sigma \to E$, d'un état initial e_0, et d'un ensemble $F \subseteq E$ d'états finaux (ou terminaux).

On appelle **calcul**, toute suite de transitions de e_0 à e_n, en ayant lu le mot m.

On appelle **calcul acceptant**, un parcours de l'automate allant de l'état initial vers un des états finaux.

Les automates

Définitions (suite)

Le **langage reconnu** (ou langage accepté) $\mathcal{L}(\mathcal{A})$, c'est l'ensemble des mots ayant un calcul acceptant.

Un **alphabet**, c'est un ensemble fini de lettres ou de symboles.

Un **mot** m est une suite finie de lettres $m = a_1 a_2 ... a_n (a_i \in \Sigma)$; de longueur n.

Le mot vide, noté ϵ, c'est le mot de longueur 0. Il n'est composé d'aucune lettre.

Les automates

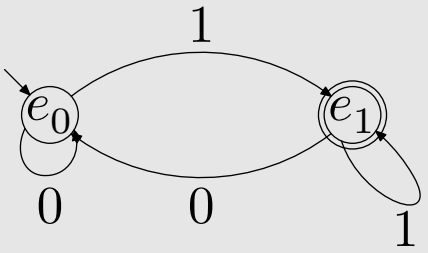
Définitions (suite)

Un état dépend uniquement :

- de l'état précédent
- du symbole lu.

Automates

Exemple d'un automate



Automates

Un autre exemple d'automate

Vérificateur de format horaire