S11 : Prolog

Enseignant : Stéphane LE PEUTREC
Assistant : Jonathan LAUPER

Instructions

- Deadline : jeudi suivant à 11:00

1. Arbre de recherche

Dessinez les arbres de recherches des questions ci-dessous :

- ?- p(X).
- ?- p(X), p(Y).
- ?- p(X), !, p(Y).

avec l'implémentation suivante pour le prédicat p :

```
p(1).
p(2) :- !.
p(3).
```

2. Prédicats avec cut

Implémentez deux versions de chacun des prédicats qui suivent. Une version sans cut et une version avec cut visant à réduire au maximum l'arbre de recherche pour une question. Utilisez le prédicat prédéfini trace/1 pour tracer les appels à vos prédicats.

- separate(+X,+L,?L1,?L2) : vrai si L1 est la liste constituée des éléments de la liste L qui sont inférieurs ou égaux à X, L2 est la liste constituée des éléments de la liste L qui sont strictement supérieurs à X.

  Exempl e : ?- separate(7,[4,9,23,2,6,11],L1,L2).
  L1=[4,2,6], L2=[9,23,11]

- myUnion(+E1,+E2,?E3) : où E1, E2 et E3 sont des listes représentant des ensembles (leurs éléments sont distincts deux à deux), myUnion(E1,E2,E3) est vrai si la liste E3 est l'union ensembliste des listes E1 et E2.
  Exemple : ?- myUnion([3,6,2],[1,9,3],E).
  E = [6,2,1,9,3]

- myIntersection(+E1,+E2,?E3) : où E1, E2 et E3 sont des listes représentant des ensembles (leurs éléments sont distincts deux à deux), myIntersection (E1,E2,E3) est vrai si la liste E3 est l'intersection des listes E1 et E2.
  Exemple : ?- myIntersection ([3,6,1,2],[1,9,3],E).
  E = [3,1]

Programmation logique

Implémentez trois versions du prédicat qui suit. Une version récursive sans cut et une version récursive avec cut et une version récurvice terminale avec accumulateurs et cut.

- maxmin(+L,?Max,?Min) où L est une liste de valeurs numériques, Max est la plus grande de ces valeurs et Min est la plus petite.
  Exemple : ?- maxmin([6,8,3,5,9,7],Max,Min)
  Max=9, Min=3

3. Correction de prédicat

Une implémentation du prédicat myMax(+X,+Y,?Z) avec le cut est proposée ci-dessous. myMax(X,Y,Z) est vrai si Z est le maximum de X et Y.

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

- Expliquez pourquoi cette implémentation n'est pas correcte. Montrez à l'aide d'un arbre de recherche que pour certaines questions elle retourne un mauvais résultat
- Proposez une correction

4. DCG et expression régulières

Implémenter des analyseurs syntaxiques à l'aide de DCG pour les expressions régulières qui suivent :

- a*bca*
- (a.|bc)*uu*

Indications : commencez par déterminer une grammaire algébrique correspondant à l'expression régulière, puis utilisez le formalisme des DCG pour implémenter un analyseur de cette grammaire.

Exemple : pour l'expression régulière 'abc', une grammaire possible est ex1 -> abc. On implémente l'analyseur syntaxique avec la règle DCG : ex1 --> [a,b,c].

Utilisez le prédicat prédéfini char_type/2. Exemple: char_type(C,ascii) vrai si C est un caratère ascii.

5. Mini-projet - Partie 1

Le but est de développer des prédicats de reconnaissance et manipulation d'un sous ensemble des expressions régulières.

Les expressions régulières traitées dans le cadre de ce mini-projet utilisent les caractères ascii et les caractères spéciaux * . ( ) | et peuvent être récursives.

Exemples :

- ab.*ba
- (ab)*cd
- a(bc*|f)*

La grammaire utilisée est la suivante :

exp → subExp '|' exp
exp →subExp
subExp → term subExp | term

Programmation logique

term → subTerm* | subTerm
subTerm→ <char> | . | ' (' exp') '

Travail à faire

Implémentez un analyseur syntaxique à l'aide de DCG pour cette grammaire et ajoutez des arguments afin de générer l'arbre syntaxique tel que décrit ci-dessous

Arbre syntaxique demandé

- la racine de l'arbre d'une expression régulière est : exp
- les sous-expressions d'une expression forment une liste
- l'arbre syntaxique respecte les conversions suivantes :

Règle	Arbre syntaxique
subTerm→<char>	char(<char>) Exemples: char(a), char(c)
subTerm→.	point
term →subTerm*	Iter(<arbre syntaxique de subTerm>) Exemples : iter(char(a)), iter(point)
exp →subExp	exp(<arbre syntaxique de subExp>)
exp → subExp '|' exp	exp(or(<arbre synt. de subExp>,<arbre synt. de exp>))

Exemples

Expression régulière	Arbre syntaxique
abc	exp([char(a), char(b), char(c)])
ab*.a	exp([char(a), iter(char(b)), point, char(a)])
(ab)*c	exp([iter(exp([char(a), char(b)])), char(c)])
ab | cd	exp(or(exp([char(a), char(b)]), exp([char(c), char(d)])))
a(bc|de)*g	exp([char(a), iter(exp(or(exp([char(b), char(c)]), exp([char(d), char(e)])))), char(g)])