Compilation - TD2

Analyse lexicale, fin
Analyse syntaxique, début

1 un AFD directement à partir d'une expression régulière

On se donne une expression régulière r ne comportant pas le symbole #, et on lui associe sa version étendue $r\#$: le caractère # est là pour marquer la fin de l'expression régulière. On se demandera à la fin pourquoi on a ajouté ce #...

On associe à cette expression régulière étendue un *arbre abstrait* défini de la manière suivante : à un symbole de l'alphabet (resp. à ϵ) correspond une feuille étiquetée avec ledit symbole (resp. ϵ) ; à un opérateur de construction des expressions régulières correspond un nœud décoré avec l'opérateur, ayant un ou deux fils suivant les cas. Par convention, on supposera que l'opérateur de concaténation associe à gauche, i.e. abc désigne $(a.b).c$.

De plus, à chaque feuille décorée avec un symbole qui n'est pas ϵ est associé un entier unique appelé *position*. On a donc une *position* pour chaque symbole de l'alphabet apparaissant dans l'expression régulière, et on choisira d'ailleurs ces entiers dans l'ordre d'apparition des symboles dans l'expression régulière.

Question 1-1 Représenter un arbre abstrait associé à l'expression régulière $(a|b)^*abb$.

On va construire, pour chaque feuille de l'arbre de position i, la fonction POSITION_SUIVANTE(i) qui donne l'ensemble des positions qui peuvent suivre la position i dans une chaîne reconnue par cet arbre.

Question 1-2 Décorez votre arbre avec POSITION_SUIVANTE, et espliquez comment en déduire un automate fini reconnaissant l'expression régulière.

Il ne reste plus qu'à contruire POSITION_SUIVANTE. Pour cela, on commence par définir, pour chaque nœud n de l'arbre, les fonctions PREMIÈRE_POSITION(n) et DERNIÈRE_POSITION(n) qui désignent respectivement l'ensemble des *positions* pouvant reconnaître les premier et dernier symboles d'une chaîne engendrée par la sous-expression de racine n. On aura également besoin d'un prédicat auxiliaire ANNULABLE(n), défini également sur l'ensemble des nœuds de l'arbre, et indiquant si un nœud n est susceptible d'engendrer la chaîne vide ϵ.

Question 1-3 Comment calculer ces fonctions sur un arbre abstrait donné ?

Question 1-4 Comment en déduire la fonction POSITION_SUIVANTE ?

2 Grammaires

Question 2-1 Quelle est la différence entre une grammaire et un ensemble de définitions régulières ?

Question 2-2 Rappelez ce qu'est une grammaire récursive à gauche, récursive à droite, avec récursion directe ou indirecte. Comparez les orientations politiques de la récursivité de la grammaire, d'une part, et de l'associativité des opérateurs impliqués, d'autre part.

Question 2-3 Soit la grammaire

$$R \rightarrow R \vee R \mid R\,R \mid R^* \mid (R) \mid a \mid b \quad .$$

Pourquoi cette grammaire est-elle ambigüe ? Proposez une grammaire équivalente non ambigüe donnant les priorités et associativités usuelles aux opérateurs (* a la plus haute priorité, ensuite vient la concaténation, enfin $\vee$, et tous ces opérateurs sont associatifs à gauche).

Question 2-4 Donnez l'arbre syntaxique et l'arbre syntaxique réduit pour

$$(aa \vee bb \vee (ab \vee ba)((aa \vee bb)) * (ab \vee ba))*$$

(vous vous souvenez ?) suivant la grammaire précédente.

Question 2-5 Écrire une grammaire non ambigüe pour le langage des parenthèses et crochets bien équilibrés, où par convention chaque crochet fermant ferme aussi toutes les parenthèses ouvertes en suspens (entre le crochet en question et le crochet ouvrant que l'on "referme").

Exemple : [((([( ])].

Question 2-6 Quel est le problème avec la grammaire suivante ?

S → 0 | A
A → A B
B → 1

Décrire un algorithme éliminant dans une grammaire les productions "fautives" au sens illustré ci-dessus.

Question 2-7 (des fois qu'il reste du temps) Donnez une grammaire pour un langage fonctionnel simplifié, et une grammaire pour un langage impératif simplifié.