Compilation - corrigé du TD3

Analyse syntaxique LL(1)

Que cela soit bien clair, tout ceci a peu d'intérêt pratique autre que de faire travailler vos neurones, puisque `yacc` est un parseur LALR. Par contre cela va motiver l'analyse LALR.

1 Analyse récursive et dérécursivation à gauche

Question 1-1 L'idée de l'analyse récursive est de traduire directement les productions de la grammaire en des fonctions booléennes qui s'appellent récursivement : la concaténation devient un *et logique*, l'alternative un *ou logique*, les symboles terminaux renvoient *vrai* en consommant un token, ou bien *faux* sans rien consommer. On peut ainsi utiliser la sémantique d'évaluation paresseuse des constructeurs booléens dans les langages impératifs.

Le rapport avec la récursion à gauche, c'est que s'il y en a elle se traduit par une boucle infinie...

Question 1-2 Le langage reconnu est $(\beta_1|\beta_2|...|\beta_n)(\alpha_1|\alpha_2|...|\alpha_n)*$ ce qui suggère immédiatement la transformation de la production $A \rightarrow A\alpha_1|\cdots|A\alpha_n|\beta_1|\cdots|\beta_n$ en

$$\begin{array}{lcl} A & \rightarrow & \beta_1 B \mid \beta_2 B \mid ... \mid \beta_n B \\ B & \rightarrow & \alpha_1 B \mid \alpha_2 B \mid ... \mid \alpha_n B \mid \epsilon \end{array}$$

Question 1-3

$$\begin{array}{lcl} A & \rightarrow & zK \mid ByK \\ K & \rightarrow & XYzK \mid zK \mid \epsilon \end{array}$$

Question 1-4 Il suffit de commencer par substituer ce qu'il faut pour que la récursivité devienne directe, et on est ramené au cas précédent.

Question 1-5

Ptite flemme...

2 Analyse prédictive et factorisation à gauche

Question 2-1 L'analyse récursive a mauvaise réputation à cause du coût du backtracking qu'elle implique. L'idée de l'analyse prédictive ou LL(1) est donc de précalculer, pour chaque alternative de règle dans la grammaire, l'ensemble First des terminaux que peut produire cette alternative en première position (cela ressemble à ce qui a été vu au TD précédent). L'analyseur est alors un ensemble de fonctions mutuellement récursives, une par règle de la grammaire, s'appelant les unes les autres en fonction du token en cours. Pour une règle $S \rightarrow A_1 \mid A_2$, avec l'alternative $A_1 = S_1 S_2$..., la fonction `analyseS` est en gros :

```
analyseS() {
   Si TokenEnCours appartient à First(A1)
   alors analyseS1; analyseS2; ...
   sinon si TokenEnCours appartient à First(A2) ...
   }
```

Le rapport avec la factorisation à gauche c'est que cela ne marche que si tout est bien factorisé à gauche : lorsque deux alternatives d'une règle ont le même terminal en première position, l'analyseur ainsi construit ne sait décider quelle alternative suivre. On appelle cela un conflit *first/first*. La solution est de factoriser ce terminal dans les deux alternatives.

Question 2-2 Lorsqu'une règle peut produire un ϵ (on dit que le non-terminal T correspondant est annulable, et cela se calcule très bien par induction comme au TD2) il faut également précalculer l'ensemble Follow des terminaux qui peuvent suivre ce non terminal dans l'ensemble de la grammaire. Attention, il y a un ensemble First par *alternative*, alors qu'il y a un ensemble Follow par *règle annulable*. En reprenant l'exemple précédent, dans lequel à présent $S_1 = S_3 \mid \epsilon$ est annulable, la fonction `analyseS1` sera :

```
analyseS1() {
   Si TokenEnCours appartient à First(S3) // on le mange
   alors analyseS3
   sinon si TokenEnCours appartient à Follow(S1) // on mange le epsilon,
   // c'est à dire on ne fait rien
   }
```

On voit qu'il y a un autre risque de conflit dit *first/follow conflict*, lorsque `analyseS1` ne peut pas décider quelle branche prendre. Un exemple est la grammaire :

$S \rightarrow Aab$
$A \rightarrow A \mid \epsilon$

et un autre exemple sera vu en 2.3. Pour l'exemple ci-dessus, on constate qu'il suffit de dérécursiver à gauche.

La morale c'est que la construction automatique de l'analyseur fonctionne uniquement pour certaines grammaires, celles qui n'ont pas de conflit LL(1), et qu'on appelle donc des grammaires LL(1) (ce n'est pas très logique). Étant donné une grammaire, il est souvent possible de la bricoler pour qu'elle devienne LL(1). Souvent ou toujours ? Encore une question pour Marianne.

Question 2-3 Conflit *first/follow* sur ElseTailOpt.

Question 2-4 Grosse flemme.