Option Traduction de Programmes
TD Mercredi 5 février

Analyse Syntaxique

1. Dans un automate déterministe il n'y a pas de transitions ϵ. Donc, le processus de calculer les ensembles prévision ne peut pas être basé sur les transitions ϵ; donner une description plus précise de comment ces ensembles sont calculés.

   Réponse Chaque fois qu'un état de l'automate déterministe contient deux items de la forme $X \to \alpha.Y\beta$ et $Y \to .\gamma$, ajouter $Prem(\beta)$ à l'ensemble de prévision de l'item $Y \to .\gamma$.
   Et, si $\beta \xrightarrow{*} \epsilon$, tout élément de l'ensemble de prévision de $X \to \alpha.Y\beta$ sera propagé à celui de $Y \to .\gamma$.
   Et enfin la règle "normale" de propagation des ensembles comme dans le cas de l'automate nondéterministe: si un état contient un item $X \to \alpha.y\beta\ \{E\}$, E est ajouté à l'ensemble de l'item $X \to \alpha y.\beta\ \{E\}$ dans le y successeur de l'état.

   En particulier, la partie des ensembles qui n'est pas propagée d'autres états, peut-elle être calculée à partir des items d'un état?
   Réponse Oui.

2. Pour les grammaires suivantes:

   - Dessiner les automates LR(1), LALR(1), SLR(1)
   - Décider si la grammaire est LR(1), LALR(1) ou SLR(1)
   - Pour les cas où la réponse est "non", donner un exemple d'un programme qui produit un conflit.

   (a) $S \to X\$$
   $X \to Ma \mid bMc \mid dc \mid bda$
   $M \to d$
   Réponse Non SLR(1); pour le programme dc, après lecture de d, faut-il réduire par $M \to d$?
   LALR(1); le d successeur de l'état initial contient les items $M \to d.\ \{a\}$ et $X \to d.c\{\$\}$; donc pas de conflit
   LR(1) parce que LALR(1)

   (b) $S \to (X \mid E] \mid F)$
   $X \to E) \mid F]$
   $E \to A$
   $F \to A$

$A \;\to$
Réponse Non SLR(1); pour le programme (), après lecture de (, on fait la réduction par $A \to \epsilon$; on a maintenant (A sur la pile; faut-il réduire par $E \;\to\; A$ ou $F \;\to\; A$?
LALR(1); le (A successeur de l'état initial contient les items $E \to A.\ \{)\}$ et $F \to A.\{]\}$; donc pas de conflit
LR(1) parce que LALR(1)

3. Pour la grammaire et le programme suivants, combien d'analyses de sous-mots seraient trouvées par un algorithme d'analyse par programmation dynamique?
$P \;\to\; Bloc$
$Bloc \;\to\; \{\ Dliste\ Iliste\ \}$
$Dliste \;\to\; D; Dliste \mid \epsilon$
$Iliste \;\to\; I; Iliste \mid \epsilon$
$D \;\to\; int\ decliste$
$decliste \;\to\; decl \mid decl, decliste$
$decl \;\to\; identificateur \mid identificateur = exp$
$I \;\to\; exp \mid identificateur \;=\; exp \mid Boucle \mid Bloc \mid \epsilon$
$Boucle \;\to\; for\ (\ I; I; I\ )\ I\ exp \;\to\; arith \mid arith\ boolop\ arith$
$arith \;\to\; M \mid M\ addop\ arith$
$M \;\to\; F \mid F\ mulop\ M$
$F \;\to\; identificateur \mid constante \mid identificateur\ postop \mid (arith)$

```
{ int x=0,y;
for (y=0;2*y<10;y++) x++;
x--;
y=0;
}
```

Réponse
Enormément!
P: un seul; tout le programme
$Bloc$: comme pour P
$Dliste$: tous les suites de longueur 0 du programme, une cinquantaine!
et la "vraie" $Dliste$ `int x=0,y;`
$Iliste$: tous les suites de longueur 0 du programme,
et `y;` et `y;for (y=0;2*y<10;y++) x++;` et `for (y=0;2*y<10;y++) x++;` et `x++;` et `y;for (y=0;2*y<10;y++) x++;x--;` et `for (y=0;2*y<10;y++) x++;x--;` et `x++;x--;` et `x--;` et `y;for (y=0;2*y<10;y++) x++;x--;y=0;` et `for (y=0;2*y<10;y++) x++;x--;y=0;` et `x++;x--;y=0;` et `x--;y=0;` et `y=0;`
etc. Notez que la grammaire est ambigue! L'analyseur va trouver les deux analyses de `y=0` comme un I.