TD analyse grammaticale

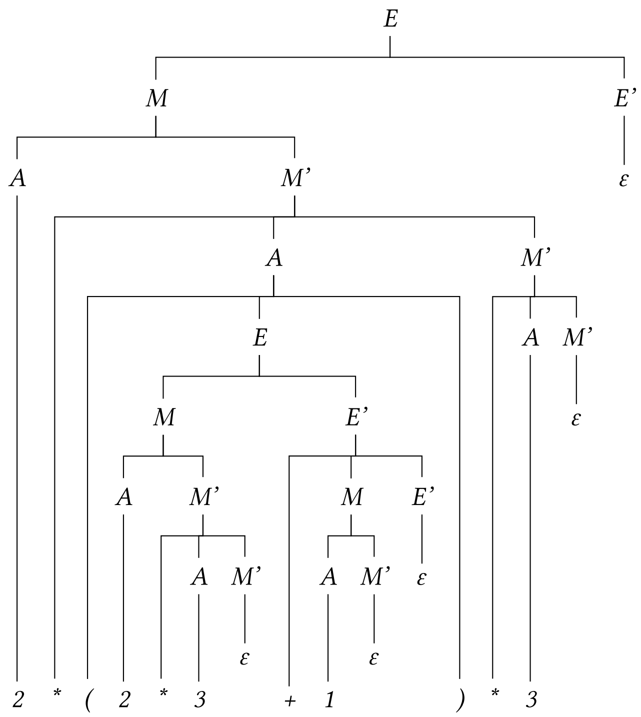
Exercice 1 Dérivation

Voici une grammaire pour les expressions arithmétiques :

$$
\begin{array}{rcl}
E & \rightarrow & M\ E' \\
E' & \rightarrow & +\ M\ E' \\
 & | & \varepsilon \\
M & \rightarrow & A\ M' \\
M' & \rightarrow & *\ A\ M' \\
 & | & \varepsilon \\
A & \rightarrow & (\ E\ ) \\
 & | & \mathsf{n}
\end{array}
$$

Donner un arbre de dérivation pour l'entrée 2 * (2 * 3 + 1) * 3.

Correction



Exercice 2 Grammaires et associativité

Le cours proposait la grammaire G_0 suivante pour les expressions arithmétiques :

$$
\begin{array}{rcl}
E & \rightarrow & E + M \\
 & | & M \\
M & \rightarrow & M * A \\
 & | & A \\
A & \rightarrow & (\ E\ ) \\
 & | & \mathsf{n}
\end{array}
$$

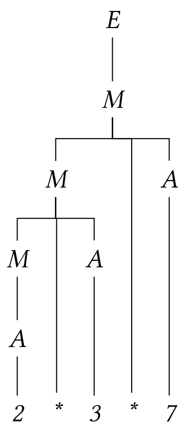
En voici plusieurs variantes :

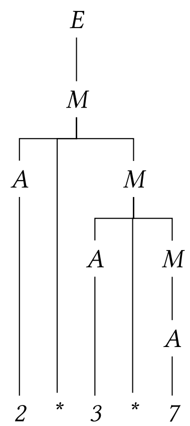
Grammaire G_1	Grammaire G_2	Grammaire G_3	Grammaire G_4
$\begin{array}{rcl} E & \to & E + E \\ & \mid & M \\ M & \to & M * M \\ & \mid & A \\ A & \to & (E) \\ & \mid & \mathsf{n} \end{array}$	$\begin{array}{rcl} E & \to & M + M \\ & \mid & M \\ M & \to & A * A \\ & \mid & A \\ A & \to & (E) \\ & \mid & \mathsf{n} \end{array}$	$\begin{array}{rcl} E & \to & M + E \\ & \mid & M \\ M & \to & A * M \\ & \mid & A \\ A & \to & (E) \\ & \mid & \mathsf{n} \end{array}$	$\begin{array}{rcl} E & \to & E + M \\ & \mid & M \\ M & \to & M * A \\ & \mid & A * A \\ & \mid & \mathsf{n} \\ A & \to & (E + M) \\ & \mid & \mathsf{n} \end{array}$

1. Deux de ces grammaires ne décrivent pas exactement le même langage que G_0. Lesquelles ? Quel langage décrivent-elles ?
2. Donner tous les arbres de dérivation possibles pour l'entrée 2 * 3 * 7 pour G_0 et les deux autres grammaires décrivant les expressions arithmétiques. En quoi l'approche de ces trois grammaires est-elle différente ?
3. Étendre ces grammaires pour qu'elles reconnaissent également les soustractions, si cela est possible.

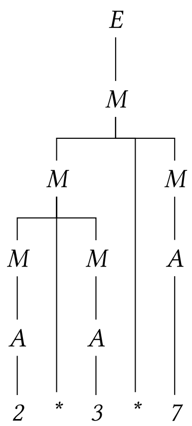
Correction

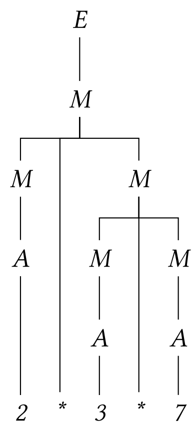
1. *Les deux grammaires décrivant un langage différent sont G_2 et G_4 :*
   - *G_2 force l'utilisation de parenthèses à chaque enchaînement de deux additions ou multiplications. Elle accepte par exemple (1+2)+3 mais pas 1+2+3.*
   - *G_4 interdit les parenthèses superflues. Elle accepte par exemple 1+2+3 mais pas (1+2)+3.*
2. *Pour G_0 et G_3, on a à chaque fois un seul arbre de dérivation possible. Ci-dessous, l'arbre pour G_0 à gauche, et celui pour G_3 à droite.*

   Pour G_1 en revanche, la grammaire permet les deux arbres de dérivation suivants.

   Interprétation :
   - *G_0 impose l'associativité à gauche : 2*3*7 est comprise comme (2*3)*7.*
   - *G_3 impose l'associativité à droite : 2*3*7 est comprise comme 2*(3*7).*
   - *G_1 est ambiguë et autorise les deux interprétations précédentes.*
3. *On peut ajouter une règle $E \to E$ - M dans G_0 ou G_1. Cela ne fonctionne en revanche pas pour G_3 (on n'admettrait pas l'entrée 1-2+3). La règle $E \to E$ - E ne convient pas, car autoriserait de mauvaises associativités (1-2+3 pouvant alors être comprise comme 1-(2+3)).*

Exercice 3 Analyse ascendante

On prend la grammaire suivante pour les expressions arithmétiques, et les règles d'analyse ascendante détaillées dans les notes de cours (tableau p.17).

$$
\begin{array}{lcl}
S & \to & E\,\# \\
E & \to & \mathsf{n} \\
 & | & E + E \\
 & | & E * E \\
 & | & (\ E\ )
\end{array}
$$

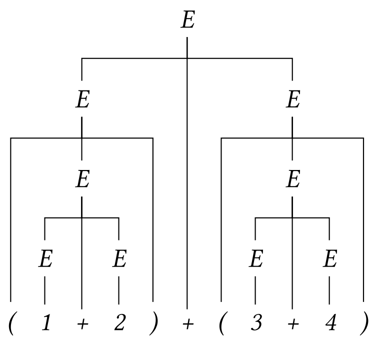
1. Détailler les étapes de l'analyse ascendante de l'entrée (1 + 2) + (3 + 4) et donner l'arbre de dérivation correspondant.
2. Détailler les étapes de l'analyse ascendante de l'entrée (1 + 2) (3). Que se passe-t-il ?

Correction

1. *On alterne étapes de progression et de réduction jusqu'à consommer toute l'entrée et réduire la pile à une unique expression E : l'analyse a réussi.*

Pile	*Entrée*	*Action*
ε	*(1+2)+(3+4)*	*S*
(	*1+2)+(3+4)*	*S*
(1	*+2)+(3+4)*	*R [E → n]*
(E	*+2)+(3+4)*	*S*
(E+	*2)+(3+4)*	*S*
(E+2	*)+(3+4)*	*R [E → n]*
(E+E	*)+(3+4)*	*R [E → E+E]*
(E	*)+(3+4)*	*S*
(E)	*+(3+4)*	*R [E → (E)]*
E	*+(3+4)*	*S*
E+	*(3+4)*	*S*
E+(	*3+4)*	*S*
E+(3	*+4)*	*R [E → n]*
E+(E	*+4)*	*S*
E+(E+	*4)*	*S*
E+(E+4	*)*	*R [E → n]*
E+(E+E	*)*	*R [E → E+E]*
E+(E	*)*	*S*
E+(E)	∅	*R [E → (E)]*
E+E	∅	*R [E → E+E]*
E	∅	*succès*

L'arbre de dérivation correspond est le suivant.



2. *La reconnaissance de cette nouvelle expression commence similairement à la précédente. En revanche, on arrive au bout d'un moment à une situation où plus aucune règle ne peut s'appliquer : l'analyse échoue, indiquant une phrase mal formée.*

Pile	*Entrée*	*Action*
ε	*(1+2)(3)*	*S*
(	*1+2)(3)*	*S*
(1	*+2)(3)*	*R [E → n]*
(E	*+2)(3)*	*S*
(E+	*2)(3)*	*S*
(E+2	*)(3)*	*R [E → n]*
(E+E	*)(3)*	*R [E → E+E]*
(E	*)(3)*	*S*
(E)	*(3)*	*R [E → (E)]*
E	*(3)*	*échec*

Exercice 4 Analyse descendante

Les règles suivantes définissent la grammaire du langage de programmation LISP.

$$
\begin{array}{lcl}
S & ::= & E\ \texttt{\#} \\
E & ::= & \texttt{sym} \\
 & | & \texttt{(}\ L\ \texttt{)} \\
L & ::= & \varepsilon \\
 & | & E\ L
\end{array}
$$

On a trois symboles non terminaux S (un programme), E (une expression) et L (une liste), et quatre symboles terminaux : les parenthèses `(` et `)`, un symbole `#` de fin d'entrée, et un symbole `sym` désignant une séquence de caractères (en pratique : un mot-clé, un opérateur, une variable...).

1. Calculer pour cette grammaire les annulables, les premiers et les suivants.
2. En déduire la table d'analyse LL(1).
3. En suivant la table précédente, analyser le programme

```
(defun double (x) (* 2 x))
```

4. Que se passe-t-il lors de l'analyse du programme suivant ?

```
(let x 3
```

Correction

1. *Annulables : seul L est annulable.*

	S	E	L
0.	F	F	F
1.	F	F	V
2.	F	F	V

Calcul des premiers : on a les équations suivantes.

$$
\begin{array}{rcl}
\mathit{Premiers}(S) & = & \mathit{Premiers}(E) \\
\mathit{Premiers}(E) & = & \{\,\texttt{sym}, \texttt{(}\,\} \\
\mathit{Premiers}(L) & = & \mathit{Premiers}(E)
\end{array}
$$

Le calcul du point fixe est rapide, comme les premiers de E se propagent directement comme premiers des autres symboles non terminaux.

	S	E	L
0.	$\emptyset$	$\emptyset$	$\emptyset$
1.	$\emptyset$	sym, (	$\emptyset$
2.	sym, (	sym, (	sym, (
3.	sym, (	sym, (	sym, (

Calcul des suivants : on a les équations suivantes.

$$
\begin{array}{rcl}
\mathit{Suivants}(S) & = & \emptyset \\
\mathit{Suivants}(E) & = & \{\,\texttt{\#}\,\} \cup \mathit{Premiers}(L) \cup \mathit{Suivants}(L) \\
\mathit{Suivants}(L) & = & \{\,\texttt{)}\,\} \cup \mathit{Suivants}(L)
\end{array}
$$

Note : l'apparition de Suivants(L) dans l'équation pour Suivants(E) vient de la règle $L \to EL$ et du fait que L est annulable. Le calcul du point fixe est à nouveau assez rapide, puisqu'il n'y a pas de dépendance cyclique.

	S	E	L
0.	$\emptyset$	$\emptyset$	$\emptyset$
1.	$\emptyset$	#, sym, (	)
2.	$\emptyset$	#, sym, (,)	)
3.	$\emptyset$	#, sym, (,)	)

2. *Table d'analyse LL(1).*

	sym	(	)	#
S	E #	E #		
E	sym	(L)		
L	$E\,L$	$E\,L$	ε	

3. *L'analyse consiste à produire une séquence de dérivation. Tant que tout se passe bien, la phrase en cours de dérivation commence par une séquence de symboles terminaux correspondant à un préfixe de la phrase à reconnaître, cette séquence étant immédiatement suivie par un premier symbole non terminal X. C'est ce symbole X qui va être expansé, en fonction du symbole terminal a trouvé dans la séquence d'entrée à cette position.*

Cible :	(defun double (x) (* 2 x)) #	X	a	*Règle*
	S	S	(	$S \to E\#$
$\to$	E #	E	(	$E \to (L)$
$\to$	(L) #	L	defun	$L \to EL$
$\to$	($E\,L$) #	E	defun	$E \to sym$
$\to$	(defun L) #	L	double	$L \to EL$
$\to$	(defun $E\,L$) #	E	double	$E \to sym$
$\to$	(defun double L) #	L	(	$L \to EL$
$\to$	(defun double $E\,L$) #	E	(	$E \to (L)$
$\to$	(defun double (L) L) #	L	x	$L \to EL$
$\to$	(defun double ($E\,L$) L) #	E	x	$E \to sym$
$\to$	(defun double (x L) L) #	L	)	$L \to \varepsilon$
$\to$	(defun double (x) L) #	L	(	$L \to EL$
$\to$	(defun double (x) $E\,L$) #	E	(	$E \to (L)$
$\to$	(defun double (x) (L) L) #	L	*	$L \to EL$
$\to$	(defun double (x) ($E\,L$) L) #	E	*	$E \to sym$
$\to$	(defun double (x) (* L) L) #	L	2	$L \to EL$
$\to$	(defun double (x) (* $E\,L$) L) #	E	2	$E \to sym$
$\to$	(defun double (x) (* 2 L) L) #	L	x	$L \to EL$
$\to$	(defun double (x) (* 2 $E\,L$) L) #	E	x	$E \to sym$
$\to$	(defun double (x) (* 2 x L) L) #	L	)	$L \to \varepsilon$
$\to$	(defun double (x) (* 2 x) L) #	L	)	$L \to \varepsilon$
$\to$	(defun double (x) (* 2 x)) #	*succès*		

4. *L'analyse de la séquence* (let x 3 *finit par échouer sur une situation pour laquelle il n'existe pas d'entrée dans la table. Cela signifie que la phrase n'est pas bien formée.*

Cible :	(let x 3 #	X	a	*Règle*
	S	S	(	$S \to E\#$
$\to$	E #	E	(	$E \to (L)$
$\to$	(L) #	L	let	$L \to EL$
$\to$	($E\,L$) #	E	let	$E \to sym$
$\to$	(let L) #	L	x	$L \to EL$
$\to$	(let $E\,L$) #	E	x	$E \to sym$
$\to$	(let x L) #	L	3	$L \to EL$
$\to$	(let x $E\,L$) #	E	3	$E \to sym$
$\to$	(let x 3 L) #	L	#	*échec*