Théorie des Langages

TD1

Expressions Régulières et Grammaires $LL(k)$

Elana Courtines
courtines.e@gmail.com
https://github.com/irinacake

Séance 1 - 13 septembre 2022
Séance 2 - 20 septembre 2022

Emmanuel Rio - emmanuel.rio@univ-tlse3.fr

1 Rappels

Écrire sous forme d'expression régulière :
- $\{a,b\}^2 \Leftrightarrow a.a + a.b + b.a + b.b$
- $L = aL + b \Leftrightarrow a^*b$ (Lemme d'Arden)
- $\lambda^n \Leftrightarrow \lambda$
- $a.(a+b) \Leftrightarrow aa + ab$

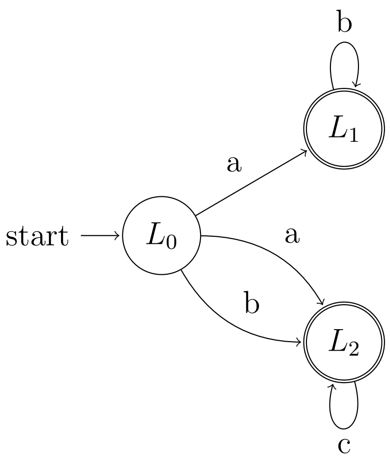
2 Analyse Lexicale

Exercice 2 :

Expression Régulière de départ :
$a.b* + (a+b).c*$

Ceci est un exemple d'une résolution menant à un automate non déterministe **à ne pas faire** :
$L_0 = a.L_1 + a.L_2 + b.L_2$
$L_1 = b* = b.L_1 + \lambda$
$L_2 = c* = c.L_2 + \lambda$

D'où l'automate :



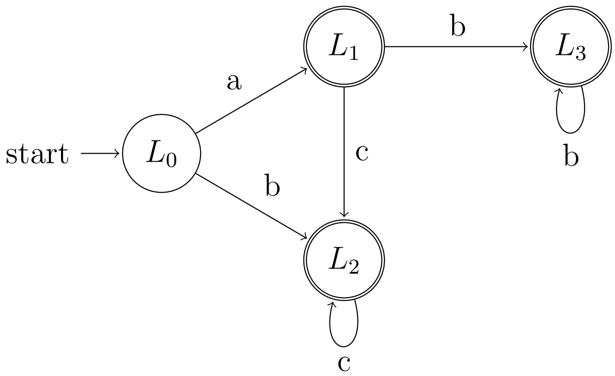
Ceci est une version plus correcte :
$L_0 = a.b* + (a + b).c*$
$L_0 = a.b* + a.c* + b.c*$
$L_0 = a.(b* + c*) + b.c*$
$L_0 = a.L_1 + b.L_2$

$L_1 = b* + c*$
$L_1 = b.b* + \lambda + c.c* + \lambda$
$L_1 = b.L3 + c.L2 + \lambda$

$L_2 = c* = c.L_2 + \lambda$

$L_3 = b* = b.L_3 + \lambda$

D'où l'automate :



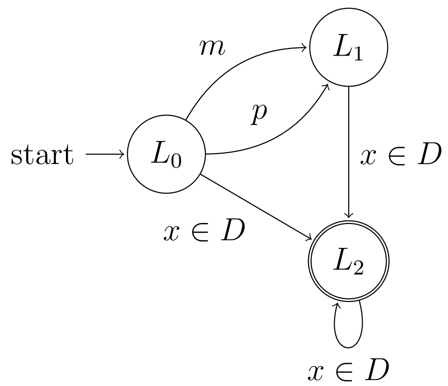
Exercice 3 :
Lexique :
- p = positif
- m = négatif
- D = ensemble des Digits
- $x \in D$ = un digit

Une Expression régulière pour représenter les entiers signés serait :

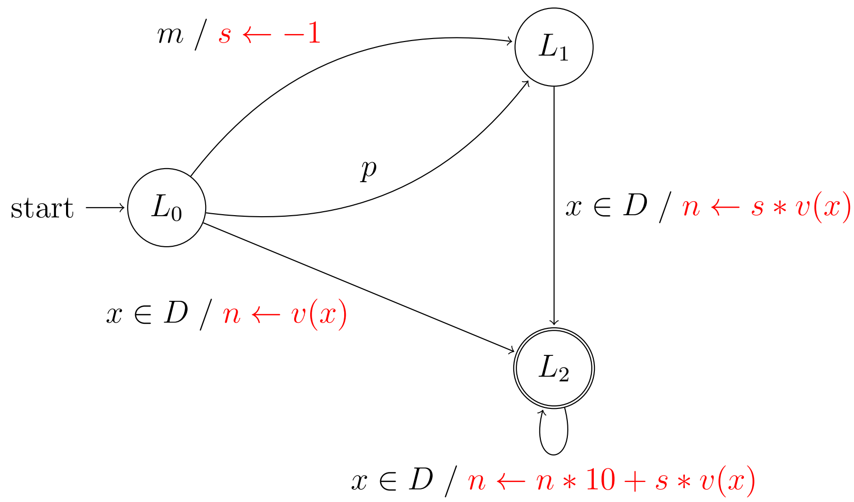
$$\begin{aligned} L_0 &= (p + n + \lambda).D.D^* \\ &= p.D.D^* + n.D.D^* + D.D^* \\ &= p.L_1 + n.L_1 + D.L_2 \end{aligned}$$

$$L_1 = D.D^* = D.L_2$$

$$\begin{aligned} L_2 &= D^* = D.D^* + \lambda \\ &= D.L_2 + \lambda \end{aligned}$$



Ajoutons maintenant les actions :
- *let* $n \leftarrow 0$ (entier)
- *let* $s \leftarrow 1$ (signe)



3 Analyse Syntaxique

Exercice 1 :

Soit la grammaire G_1 définie comme suit :
(0) - $S' \to S\ \$^k$
(1) - $S \to a\ S\ b\ S$
(1) - $S \to \lambda$

Avant même de vérifier si une grammaire est $LL(1)$, on peut essayer de regarder si elle est $LL(0)$.

Rappel :
Une grammaire $LL(0)$ est une grammaire pour laquelle il n'y a aucun choix sur les dérivations, par exemple :
- $S' \to S\$$
- $S \to aA$
- $A \to bB$
- $B \to \lambda$
Formera obligatoirement et uniquement le mot $\{ab\$\}$.

Pour en revenir sur G_1, il y a deux règles de productions pour S donc celle-ci n'est pas $LL(0)$.

G_1 est-elle $LL(1)$? Pour cela, on teste les 1-lookahead de toutes les règles de production afin de déterminer si il y a des intersections non vides.

1-lookahead($S' \to S\$$)
$= first_1(S\$\ \cancel{follow_1(S')})$
$follow_1$ est inutile ici car on a la garantie de tomber sur \$.
$= first_1(aSbS\$\ \cancel{follow_1(S')}) \ \cup\ first_1(\lambda\$\ \cancel{follow_1(S')})$
$= \{a\} \cup \{\$\}$
$= \{a, \$\}$

1-lookahead($S \to aSbS$)
$= first_1(aSbS\ \ \cancel{follow_1(S)})$
$= first_1(aSbS)$
$= \{a\} = I_1$

1-lookahead($S \to \lambda$)
$= first_1(\lambda\ follow_1(S))$
$= follow_1(S)$
$= first_1(\$\ \cancel{follow_1(S')})$ $\leftarrow$ ce qui suit S dans la partie droite de la règle (0)
$\cup\ first_1(bS\ \cancel{follow_1(S)})$ $\leftarrow$ ce qui suit le premier S dans la partie droite de règle (1)
$\cup\ first_1(follow_1(S))$ $\leftarrow$ ce qui suit le deuxième S dans la partie droite de règle (1)
$= \{\$\} \cup \{b\} \cup \cancel{follow_1(S')}$ $\leftarrow$ point fixe, on peut donc l'ignorer.
$= \{\$, b\} = I_2$

Comme $I_1 \cap I_2 = \varnothing$, on peut affirmer que G_1 est $LL(1)$.

Pour en extraire la table, il suffit de suivre ces étapes :

1. Ajouter une colonne pour chaque terminal (une table issue d'un 2-*lookahead* aurait nécessité une colonne pour chaque combinaison $x.x$ *où* $x \in alphabet$) :

a	b	$

2. Ajouter une ligne pour chaque non-terminal :

	a	b	$
S'			
S			

3. Pour chaque $1 - lookahead$, vérifier quel(s) terminal(aux) est(sont) obtenu(s), et ajouter la règle correspondante dans la case sur la ligne du non-terminal associé pour tous ces terminaux.

- $1\text{–}lookahead(S' \rightarrow S\$) = \{a, \$\}$, il faut donc ajouter la règle (0) pour les terminaux a et $\$$ sur la ligne du non-terminal S' :

	a	b	$
S'	S$(0)	err	S$(0)
S			

- $1\text{–}lookahead(S \rightarrow aSbS) = \{a\}$, il faut donc ajouter la règle (1) pour le terminal a sur la ligne du non-terminal S :

	a	b	$
S'	S$(0)	err	S$(0)
S	aSbS(1)		

- $1\text{–}lookahead(S \rightarrow \lambda) = \{b, \$\}$, il faut donc ajouter la règle (2) pour les terminaux b et $\$$ sur la ligne du non-terminal S :

	a	b	$
S'	S$(0)	err	S$(0)
S	aSbS(1)	λ(2)	λ(2)

Parse du mot *abab*\$:

mot	pile	action
abab\$	S'	(0)
abab\$	S\$	(1)
abab\$	aSbS\$	pop(a)
bab\$	SbS\$	(2)
bab\$	bS\$	pop(b)
ab\$	S\$	(1)
ab\$	aSbS\$	pop(a)
b\$	SbS\$	(2)
b\$	bS\$	pop(b)
\$	S\$	(2)
\$	\$	pop(\$)
		accept

Exercice 2 :

Soit la grammaire G_2 définie comme suit :

(0) - $S' \to S\$^k$

(1) - $S \to bRS$

(2) - $S \to RcSa$

(3) - $S \to \lambda$

(4) - $R \to acR$

(5) - $R \to b$

1. LL(0) ? Non car R et S ont plusieurs dérivations.

2. LL(1) ?

- 1-$lookahead(S \to bRS) = \{b\} = I_1$
- 1-$lookahead(S \to RcSa)$

$$\begin{aligned} &= first_1(RcSa\ follow_1(S)) \\ &= first_1(acRcSa\_\_) \cup first_1(bcSa\_\_) \\ &= \{a, b\} = I_2 \end{aligned}$$

On remarque que $I_1 \cap I_2 = \{b\} \neq \varnothing$, la Grammaire G_2 n'est donc **pas** $LL(1)$.

3. LL(2) ?

Pour S :

- 2-$lookahead(S \to bRS)$

$$\begin{aligned} &= first_2(bRS\ follow_2(S)) \\ &= first_2(bacRS\_\_) \cup first_2(bbS\_\_) \\ &= \{ba, bb\} = I_1 \end{aligned}$$

- 2-$lookahead(S \to RcSa)$

$$\begin{aligned} &= first_2(RcSa\ follow_2(S)) \\ &= first_2(acRcSa\_\_) \cup first_2(bcSa\_\_) \\ &= \{ac, bc\} = I_2 \end{aligned}$$

- 2-$lookahead(S \to \lambda)$

$$\begin{aligned} &= follow_2(S) \\ &= first_2(\$\$) \cup \cancel{first_2(follow_2(S))} \cup first_2(a\ follow_2(S)) \\ &= \{\$\$\} \cup a\,.\,follow_1(S) \\ &= \{\$\$\} \cup a\,.\,(first_1(\$) \cup \cancel{first_1(follow_1(S))} \cup first_1(a\_\_)) \\ &= \{\$\$, a\$, aa\} = I_3 \end{aligned}$$

On remarque que :

- $I_1 \cap I_2 = \varnothing$
- $I_1 \cap I_3 = \varnothing$
- $I_2 \cap I_3 = \varnothing$

Pour R :

- 2-*lookahead*$(R \to acR) = first_2(acR\_\_) = \{ac\} = I_4$
- 2-*lookahead*$(R \to b)$

$$\begin{aligned}
&= first_2(b\ follow_2(R)) \\
&= b\,.\,first_1(follow_1(R)) \\
&= b\,.\,first_1(S\ follow_1(S)) \cup first_1(cSa\ follow_1(S)) \cup \cancel{first_1(\,.\,follow_1(R))}\ ) \\
&= b\,.\,(\{b, a, \$\} \cup \{c\}) \\
&= \{bb, ba, b\$, bc\} = I_5
\end{aligned}$$

On remarque que $I_4 \cap I_5 = \varnothing$, la Grammaire G_2 est donc bien $LL(2)$.

Pour déterminer la table d'analyse, il nous faut également le 2-*lookahead*(S') :

2-lookahead$(S' \to S\$^2)$

$$\begin{aligned}
&= first_2(S\$\$\ follow_2(S')) \\
&= first_2(bRS\$\$...) \cup first_2(RcSa\$\$...) \cup first_2(\$\$...) \\
&= b * first_1(RS\$\$...) \cup first_2(acRcSa\$\$...) \cup first_2(bcSa\$\$...) \cup \{\$\$\} \\
&= b * (first_1(acRS\$\$...) \cup first_1(bS\$\$...)) \cup \{ac\} \cup \{bc\} \cup \{\$\$\} \\
&= \{ba, bb, ac, bc, \$\$\}
\end{aligned}$$

En résulte la table d'analyse de G_2 :

	aa	ac	a\$	ba	bb	bc	b\$	\$\$
S'	err	S\$\$(0)	err	S\$\$(0)	S\$\$(0)	S\$\$(0)	err	S\$\$(0)
S	λ(3)	RcSa(2)	λ(3)	bRS(1)	bRS(1)	RcSa(2)	err	λ(3)
R	err	acR(4)	err	b(5)	b(5)	b(5)	b(5)	err

Parse du mot $acb\$^k$:

mot	pile	action
acb\$\$	S'	(0)
acb\$\$	S\$\$	(2)
acb\$\$	RcSa\$\$	(4)
acb\$\$	acRcSa\$\$	pop(a),pop(c)
b\$\$	RcSa\$\$	(5)
b\$\$	bcSa\$\$	pop(b)
\$\$	cSa\$\$	err

Parse du mot $bcbcaa\$^k$:

mot	pile	action
bcbcaa\$\$	S'	(0)
bcbcaa\$\$	S\$\$	(2)
bcbcaa\$\$	RcSa\$\$	(5)
bcbcaa\$\$	bcSa\$\$	pop(b),pop(c)
bcaa\$\$	Sa\$\$	(2)
bcaa\$\$	RcSaa\$\$	(5)
bcaa\$\$	bcSaa\$\$	pop(b),pop(c)
aa\$\$	Saa\$\$	(3)
aa\$\$	aa\$\$	pop(a),pop(a)
\$\$	\$\$	pop(\$),pop(\$)
		accept

Exercice 4 :

Soit la grammaire G_4 définie comme suit :

(0) - $S' \to S\$^k$

(1) - $S \to aA$

(2) - $A \to bB$

(3) - $B \to c$

Cette grammaire est LL(0), car il n'y a qu'une seule règle de dérivation par non terminal.

Table d'analyse :

	a	b	c
S'	S(0)	S(0)	S(0)
S	aA(1)	err	err
A	err	bB(2)	err
B	err	err	c(3)

version simple :

	λ
S'	S(0)
S	aA(1)
A	bB(2)
B	c(3)

Exercice 5 :

Soit la grammaire G_4 définie comme suit :

(0) $< program > \to\ program < declList > begin < instList > end$

(1) $< instList > \to < inst >$

(2) $\to < instList > ;\ < inst >$

(3) $< inst > \to\ if < exp > then < instList > else < instList > endif$

(4) $\to\ while < exp > loop < instList > endloop$

(5) $\to\ repeat < exp > until < instList > endloop$

(6) $\to\ ID := < exp >$

(..) $< declList > \to\ ...$

(..) $< exp > \to\ ...$

Cette Grammaire **ne peut pas être** $LL(k)$, parce que $< instList > \to < instList > ; < inst >$ présente une récursivité à gauche, il faut donc l'éliminer (transformation en de nouvelles règles équivalentes mais non récursives à gauche)

On peut réécrire la règle comme suit :
$A \to B$
$A \to AcB$ où $c =$;

Autrement dit :

$L(A) =$

$$\begin{aligned} & L(B) + L(A)\,c\,L(B) \\ = \; & L(A)\,c\,L(B) + L(B) \\ = \; & L(B)(c\,L(B))* \\ = \; & (L(B)\,c)*L(B) \\ = \; & L(B)\,c\,L(A) + L(B) \\ = \; & L(B)(c\,L(A) + \lambda) \end{aligned}$$

Définissons $L(C) = c\,L(A) + \lambda$, alors :

- $A \to BC$
- $C \to c\,A$
- $C \to \lambda$

Que l'on peut alors réécrire comme :

- $< instList > \quad \to < inst >< instListOpt >$
- $< instListOpt > \to ; < instList >$
- $< instListOpt > \to \lambda$