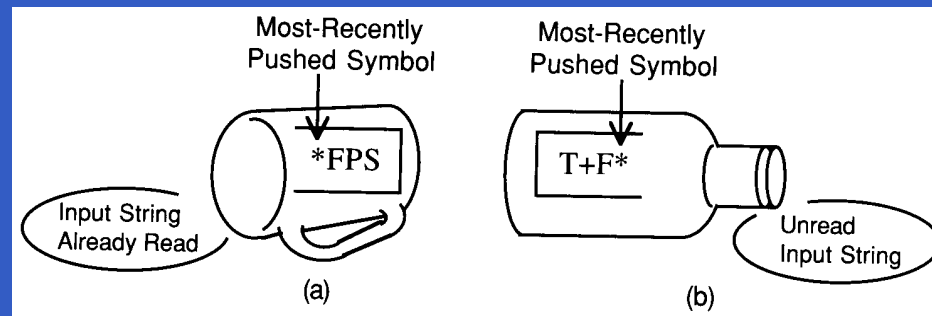


Les analyseurs LR(k) ($k = 0, 1, \dots$)



INFO010 – Théorie des langages – Partie pratique

Sébastien COLLETTE et Gilles GEERAERTS

Introduction

- **Principe de l'analyseur LR** : réduire une série de terminaux et de non-terminaux préalablement *pushés* sur le *stack* en un non-terminal.
 - En quelque sorte, on suit les règles « à l'envers ».
 - La partie droite d'une règle qui servira à la réduction est appelée *handle*.

Exemple

$$S' \rightarrow S\$$$

$$S \rightarrow Saa$$

$$S \rightarrow a$$

$$S \rightarrow \varepsilon$$

Pile	Entrée	Action	Sortie
\perp	$aa\$$	R_3	
$\perp S$	$aa\$$	S	3
$\perp Sa$	$a\$$	S	3
$\perp Saa$	$\$$	R_1	3
$\perp S$	$\$$	S	31
$\perp S\$$	ε	Accept	31

On voit que le choix des actions n'est pas trivial...

Grammaire LR(k)

Définition : Soit une grammaire $G = (V, T, P, S)$. Prenons-en la version augmentée : $G' = (V', T, P', S')$. On dit que G est LR(k) pour $k \geq 0$ si les trois conditions :

1. $S' \xRightarrow{*}_{G'} \alpha Aw \Rightarrow_{G'} \alpha \beta w,$

2. $S' \xRightarrow{*}_{G'} \gamma Bx \Rightarrow_{G'} \alpha \beta y,$

3. $\text{First}^k(w) = \text{First}^k(y)$

impliquent que : $\alpha Ay = \gamma Bx$ (c'est-à-dire $\alpha = \gamma$, $A = B$ et $y = x$).

Construction de l'analyseur LR(0)

- On va construire un **automate canonique** qui reflète les décisions prises par l'analyseur ;
- Chaque état contient plusieurs *items* : des règles dans lesquelles on a introduit des • qui symbolisent **l'avancement de l'analyseur** ;
 - Une partie de ces *items* constitue le **noyau** ;
 - Les autres sont obtenues par **fermeture** ;
- Cet automate permettra de construire les tables utilisées par l'analyseur.

Exemple – 1

Soit la Grammaire :

$$(0) \quad S' \rightarrow S\$$$

$$(1) \quad S \rightarrow (L)$$

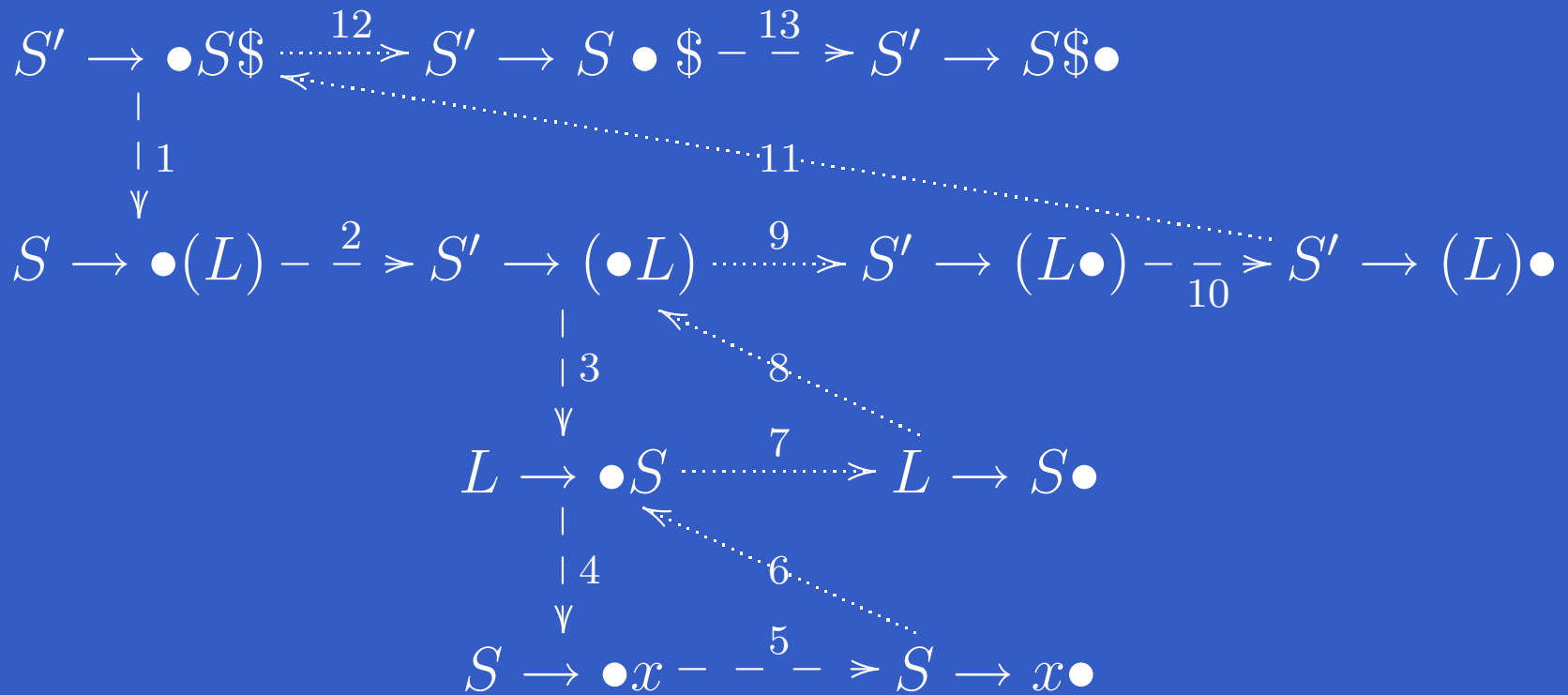
$$(2) \quad S \rightarrow x$$

$$(3) \quad L \rightarrow S$$

$$(4) \quad L \rightarrow L, S$$

Extrait de : *Modern compiler implementation in Java*, A. W. Appel

Exemple – 2 : Reconnaître (x) ...



Le \bullet représente l'état d'avancement de l'analyseur.

Exemple – 2

- On veut reconnaître un mot que l'on peut dériver de S'

Exemple – 2

$S' \rightarrow \bullet S\$$

- On veut reconnaître un mot que l'on peut dériver de S'
- Il faut donc avaler $S\$$...

Exemple – 2

$S' \rightarrow \bullet S\$$

Noyau

Le \bullet représente l'état d'avancement de l'analyseur.

- On veut reconnaître un mot que l'on peut dériver de S'
- Il faut donc avaler $S\$$...

Exemple – 2

$S' \rightarrow \bullet S\$$

Noyau

Le \bullet représente l'état d'avancement de l'analyseur.

- On veut reconnaître un mot que l'on peut dériver de S'
- Il faut donc avaler $S\$$...
- Mais S n'est **pas un terminal**!

Exemple – 2

$$S' \rightarrow \bullet S\$$$

Fermeture

Le \bullet représente l'état d'avancement de l'analyseur.

- On veut reconnaître un mot que l'on peut dériver de S'
- Il faut donc avaler $S\$$...
- Mais S n'est **pas un terminal**!

Exemple – 2

$S' \rightarrow \bullet S \$$
$S \rightarrow \bullet (L)$
$S \rightarrow \bullet x$

Fermeture

Le \bullet représente l'état d'avancement de l'analyseur.

- On veut reconnaître un mot que l'on peut dériver de S'
- Il faut donc avaler $S \$$...
- Mais S n'est **pas un terminal** !
- Reconnaître S revient à avaler un $($ ou un x .

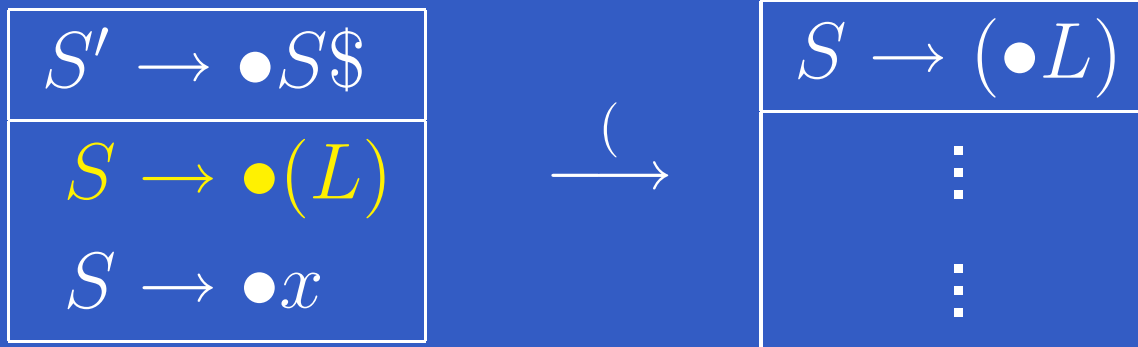
Exemple – 3

Transitions

$S' \rightarrow \bullet S\$$
$S \rightarrow \bullet (L)$
$S \rightarrow \bullet x$

Exemple – 3

Transitions



Exemple – 3

Transitions

$S' \rightarrow \bullet S \$$
$S \rightarrow \bullet (L)$
$S \rightarrow \bullet x$

$\xrightarrow{(\text{)}}$

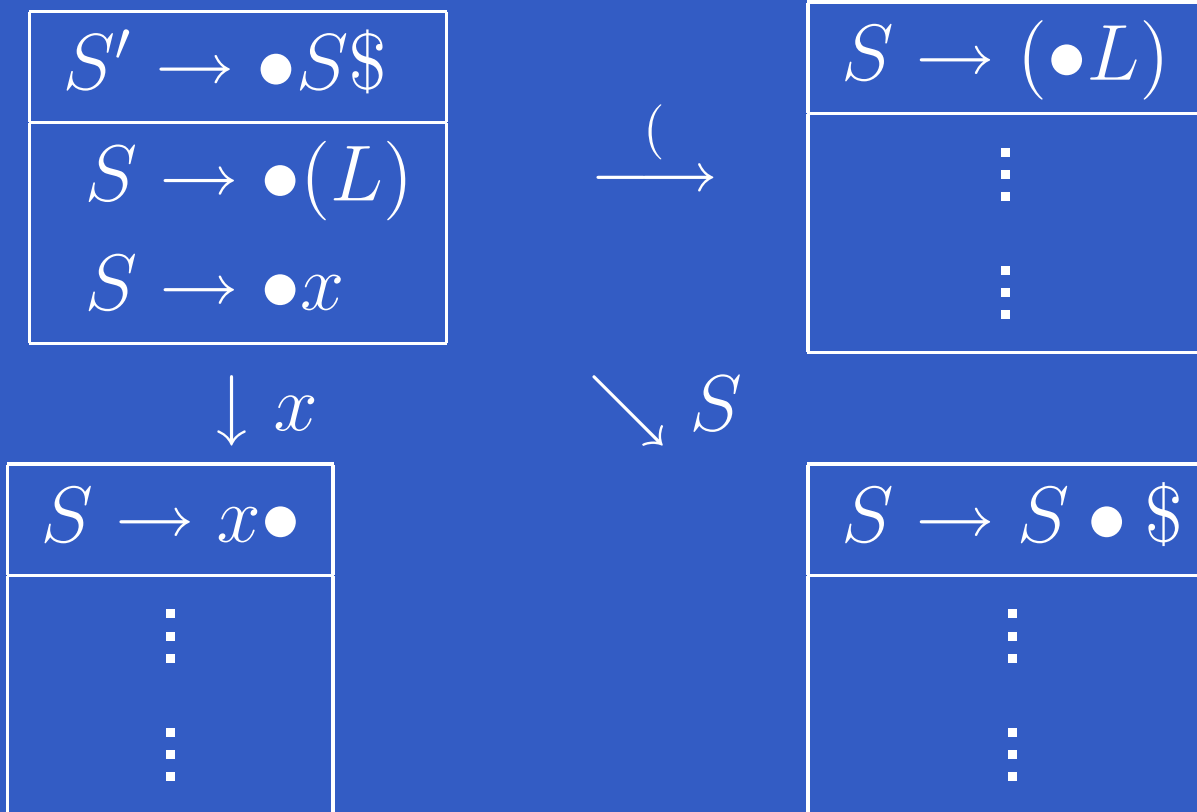
$S \rightarrow (\bullet L)$
\vdots
\vdots

$\downarrow x$

$S \rightarrow x \bullet$
\vdots
\vdots

Exemple – 3

Transitions



Exemple – 4



Noyau

- On veut reconnaître un mot que l'on peut dériver de L

Exemple – 4

$$S \rightarrow (\bullet L)$$

$$L \rightarrow \bullet S$$

$$L \rightarrow \bullet L, S$$

Fermeture (1)

- On veut reconnaître un mot que l'on peut dériver de L
- Il faut donc avaler L ou $S\$...$

Exemple – 4

$$S \rightarrow (\bullet L)$$

$$L \rightarrow \bullet S$$

$$L \rightarrow \bullet L, S$$

$$S \rightarrow \bullet (L)$$

$$S \rightarrow \bullet x$$

Fermeture (2)

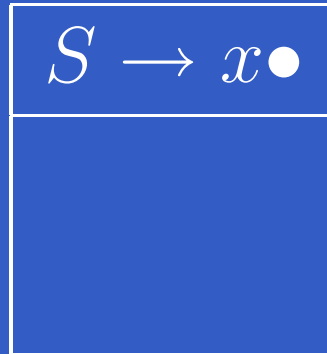
- On veut reconnaître un mot que l'on peut dériver de L
- Il faut donc avaler L ou $S\$...$
- On refait donc une étape de fermeture !

Exemple – 5

$S \rightarrow x \bullet$
\vdots
\vdots

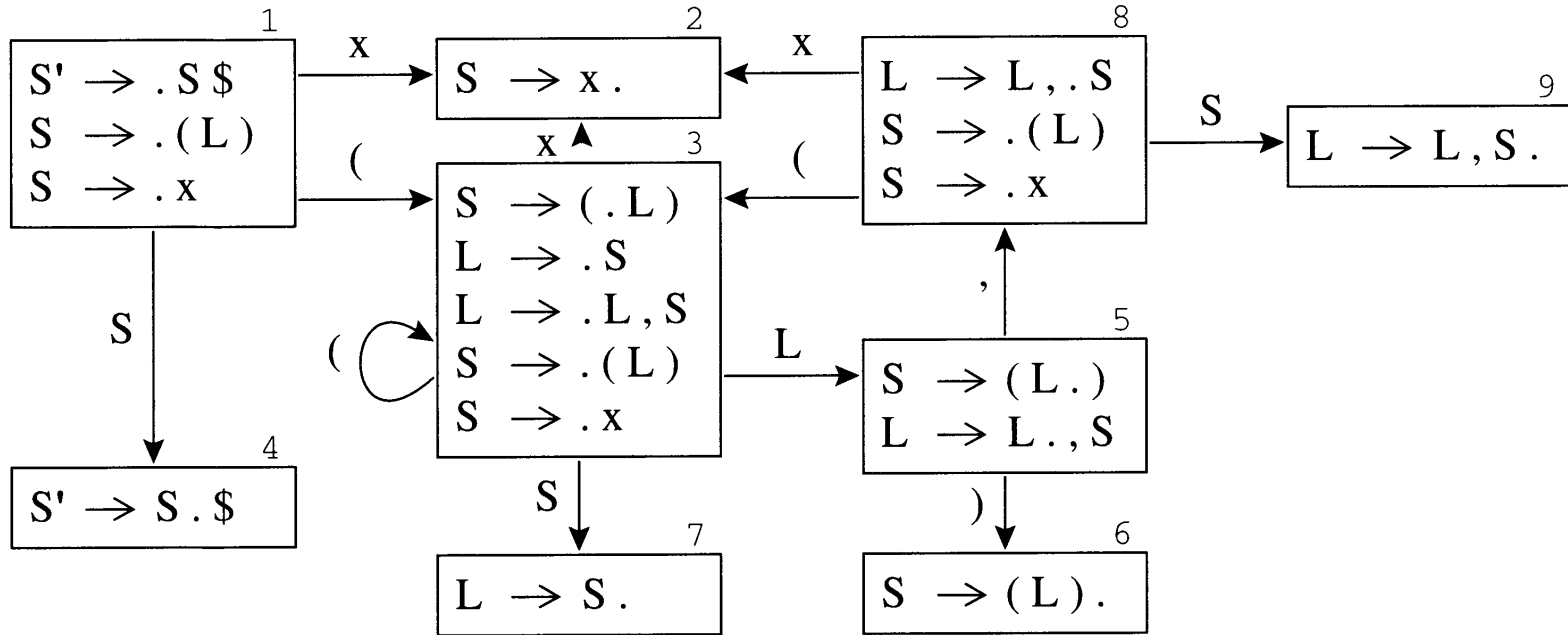
- Dans ce cas-ci, il n'y a rien à rajouter pour la fermeture !
- On a donc reconnu $S \dots$
- L'analyseur devra faire un *Reduce*.

Exemple – 5



- Dans ce cas-ci, il n'y a rien à rajouter pour la fermeture !
- On a donc reconnu S ...
- L'analyseur devra faire un *Reduce*.

Exemple – 6



Exemple – 7

État	Action
1	Shift
2	Reduce
3	Shift
4	Accept
5	Shift

État	Action
6	Reduce
7	Reduce
8	Shift
9	Reduce

Analyseur LR(0) – CFSM – 1

Fermeture(I) **begin**

répéter

$I' \leftarrow I$;

pour chaque *item* $[A \rightarrow \alpha \bullet B\beta] \in I, B \rightarrow \gamma \in G'$ **faire**

$I \leftarrow I \cup [B \rightarrow \bullet\gamma]$;

jusqu'à $I' = I$;

 return(I) ;

end

Transition(I, X) **begin**

 return(Fermeture($\{[A \rightarrow \alpha X \bullet \beta] \mid [A \rightarrow \alpha \bullet X\beta] \in$

$I\}\}))$;

end

Analyseur LR(0) – CFSM – 2

```
Items( $G'$ ) begin  
   $C \leftarrow \text{Fermeture}(\{[S' \rightarrow \bullet S]\})$  ;  
  répéter  
     $C' \leftarrow C$  ;  
    pour chaque  $I \in C, X \in T' \cup V'$  faire  
       $\lfloor C' \leftarrow C' \cup \text{Transition}(I, X)$  ;  
    jusqu'à  $C' = C$  ;  
end
```

Analyseur LR(0) – Table des actions

On construit la tables des actions comme suit :

pour chaque état s du CFMS faire

si s contient $A \rightarrow \alpha \bullet a\beta$ alors

└ Action[s] ← Action[s] ∪ Shift ;

sinon si s contient $A \rightarrow \alpha \bullet$, qui est la règle i alors

└ Action[s] ← Action[s] ∪ Reduce _{i} ;

sinon si s contient $S' \rightarrow S\$ \bullet$ alors

└ Action[s] ← Action[s] ∪ Accept ;

Exercice 1

$$(0) \quad S' \rightarrow S\$$$

$$(1) \quad S \rightarrow aCd$$

$$(2) \quad S \rightarrow bD$$

$$(3) \quad S \rightarrow Cf$$

$$(4) \quad C \rightarrow eD$$

$$(5) \quad C \rightarrow Fg$$

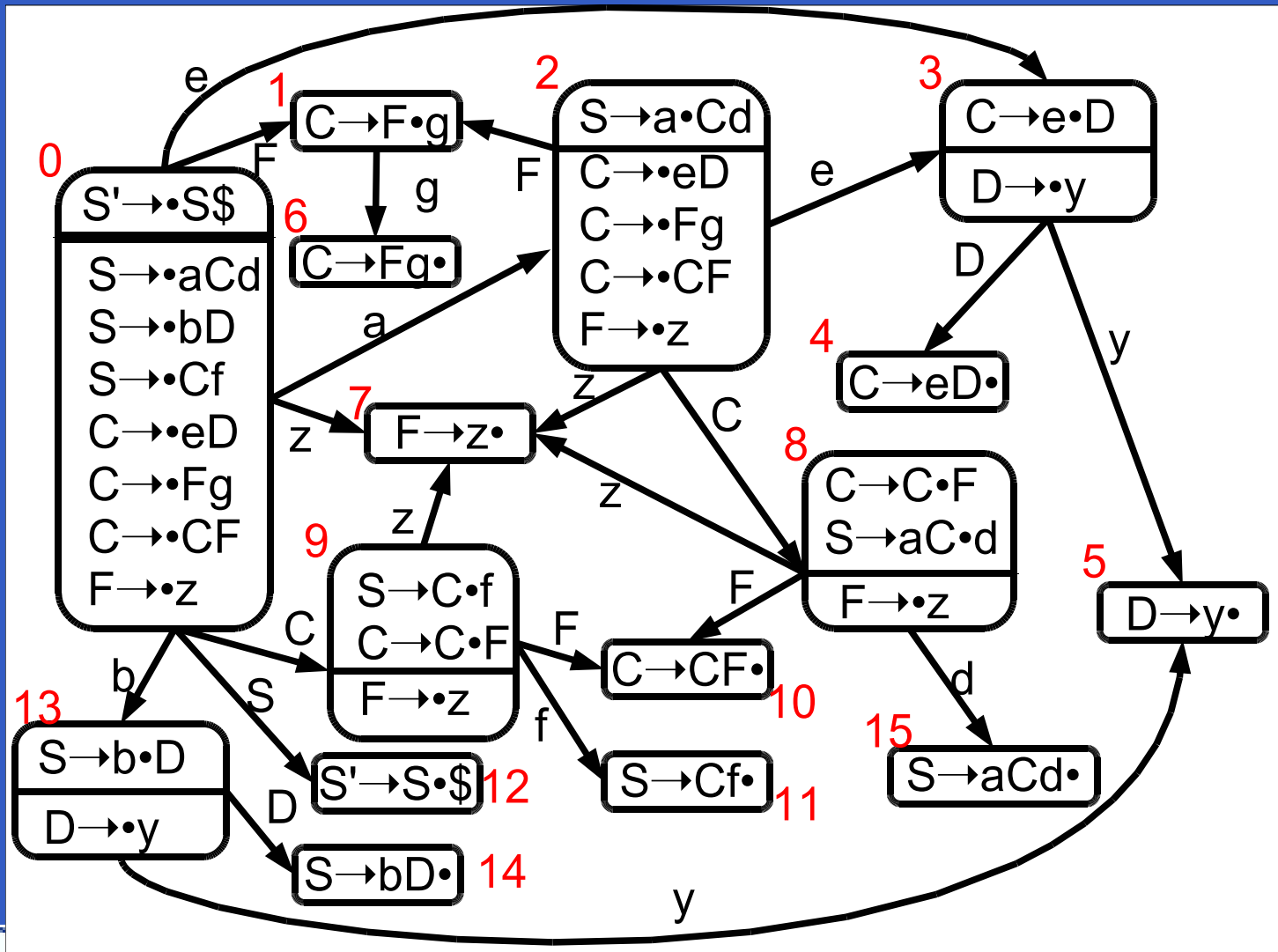
$$(6) \quad C \rightarrow CF$$

$$(7) \quad F \rightarrow z$$

$$(8) \quad D \rightarrow y$$

- Construisez l'automate canonique, et la table des actions.

Exercice 1 – Solution



Exercice 1 – Solution

État	Action	État	Action
0	Shift	8	Shift
1	Shift	9	Shift
2	Shift	10	Reduce
3	Shift	11	Reduce
4	Reduce	12	Accept
5	Reduce	13	Shift
6	Reduce	14	Reduce
7	Reduce	15	Reduce

Analyseur LR(0) – Algorithme

- L'analyseur utilise un *stack* sur lequel il *shifte* les symboles ainsi que le numéro d'état courant.
- Cela permet de revenir dans le bon état quand on fait un *reduce*.
- La *string* avalée est acceptée quand on arrive dans l'état final (dont l'action est *Accept*).
- On représente une configuration de l'analyseur par un triplet $\langle \text{stack, input, output} \rangle$.
- Initialement : $\langle \vdash 0, \omega, \varepsilon \perp \rangle$

Analyseur LR(0) – Transitions

begin

Ayant $\langle \vdash \gamma s, ax, y \perp \rangle$:

si Action[s] = Shift **alors**

└ on va en $\langle \vdash \gamma s a \text{Successeur}[s, a], x, y \perp \rangle$;

sinon si Action[s] = Reduce_j par $A \rightarrow \alpha$ **alors**

└ Ayant $\langle \vdash \gamma s' x_1 s_1 x_2 s_2 \dots x_n s, x, y \perp \rangle$ et

$\alpha = x_1 x_2 \dots x_n$, on va en :

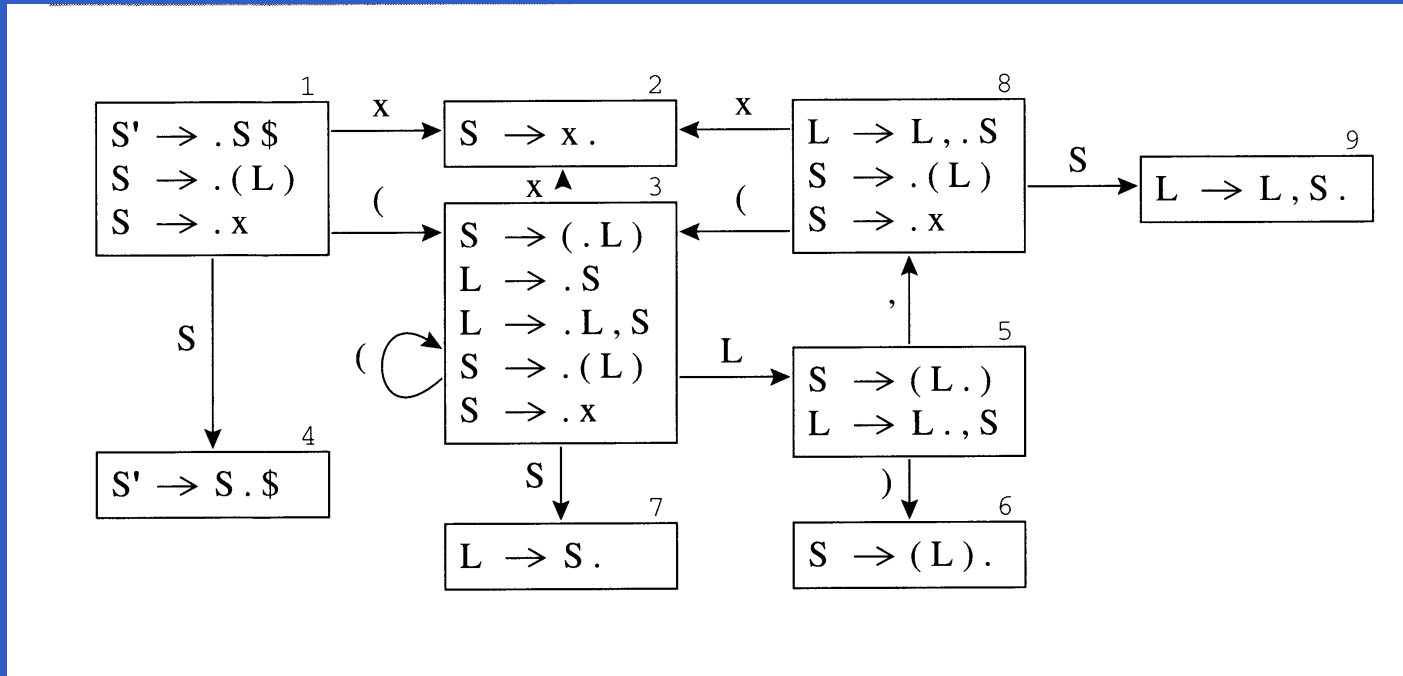
└ $\langle \vdash \gamma s' A \text{Successeur}[s', A], x, j y \perp \rangle$;

sinon si Action[s] = Accept **alors** return(OK) ;

sinon return(Error) ;

end

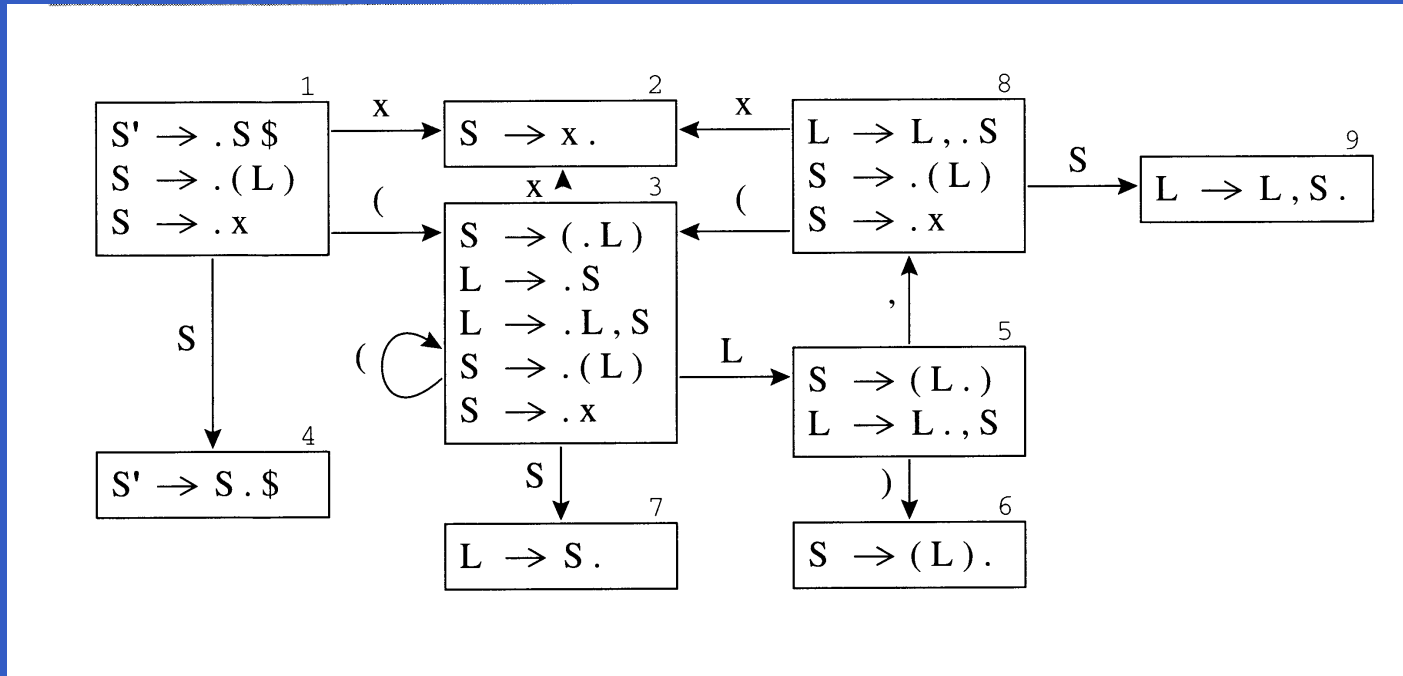
Exemple : reconnaître (x)



Config.: $\langle 1, (x)\$, \rangle$

Action: Shift

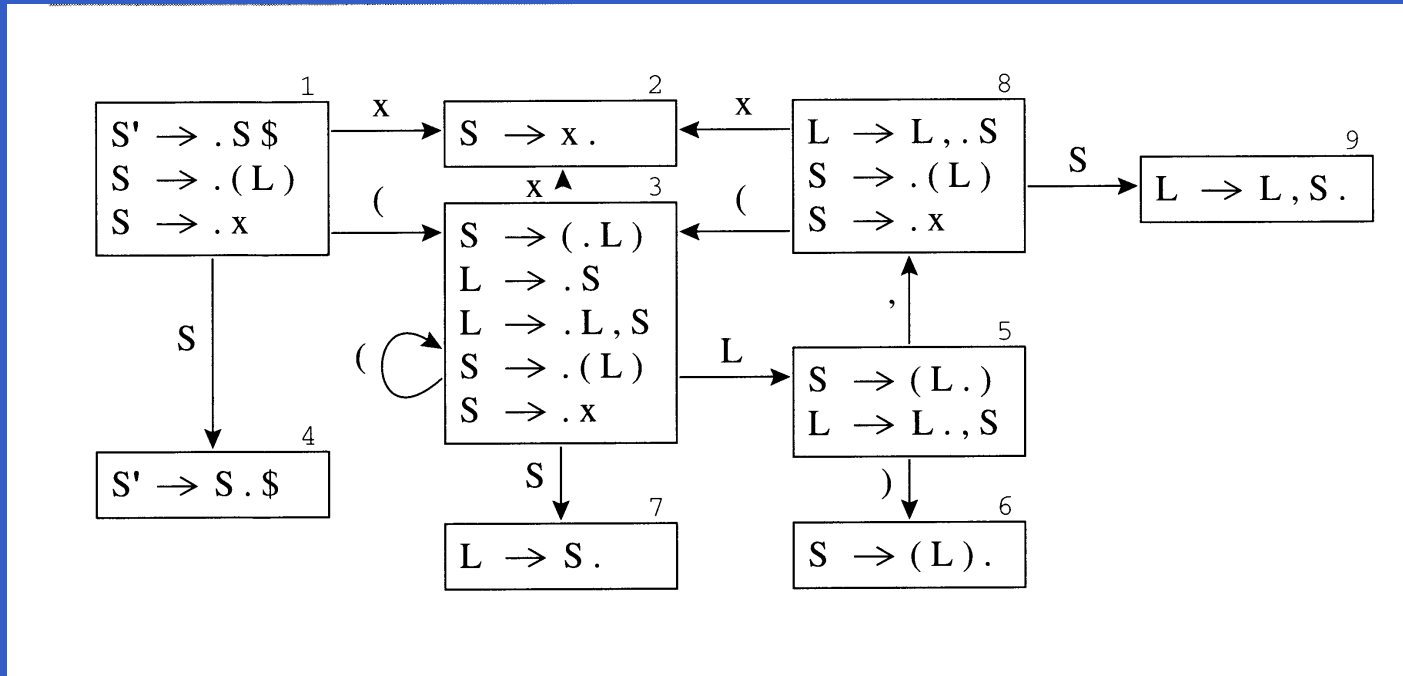
Exemple : reconnaître (x)



Config.: $\langle 1(3, x) \$, \rangle$

Action: Shift

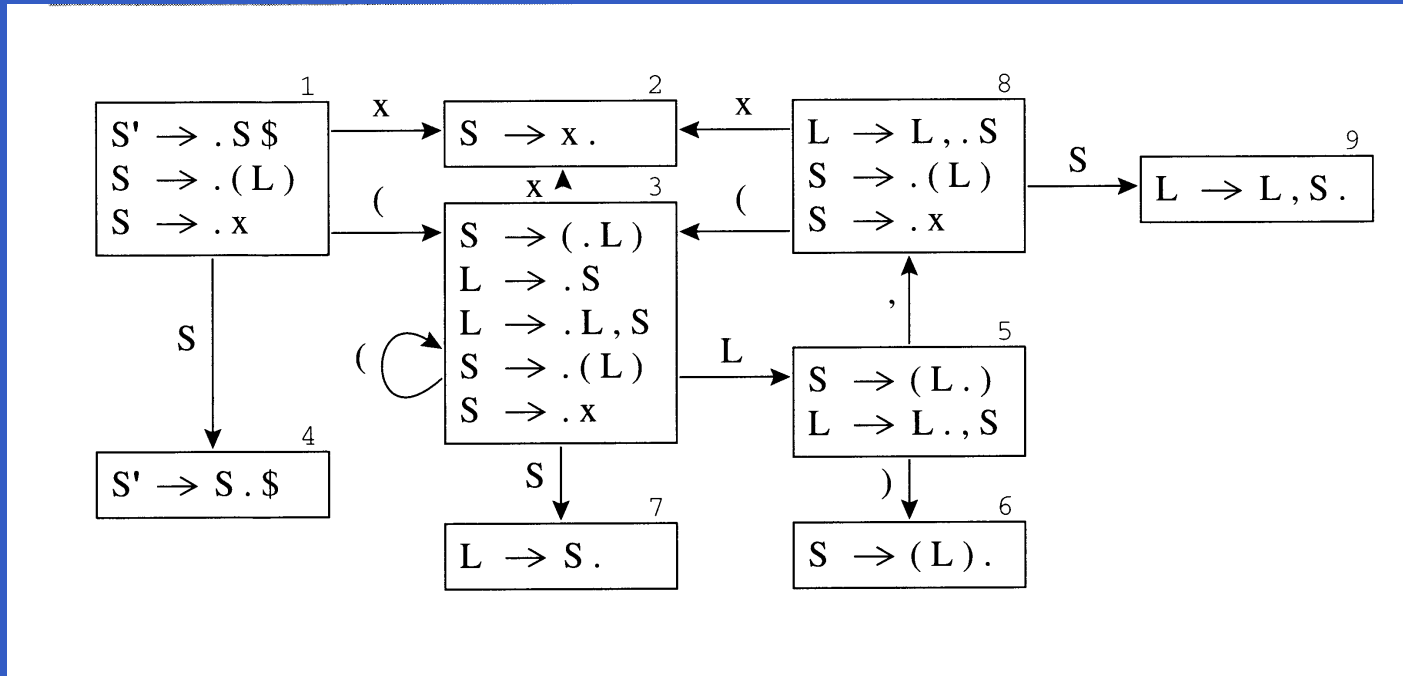
Exemple : reconnaître (x)



Config.: $\langle 1(3x2,) \$, \rangle$

Action: Reduce 2

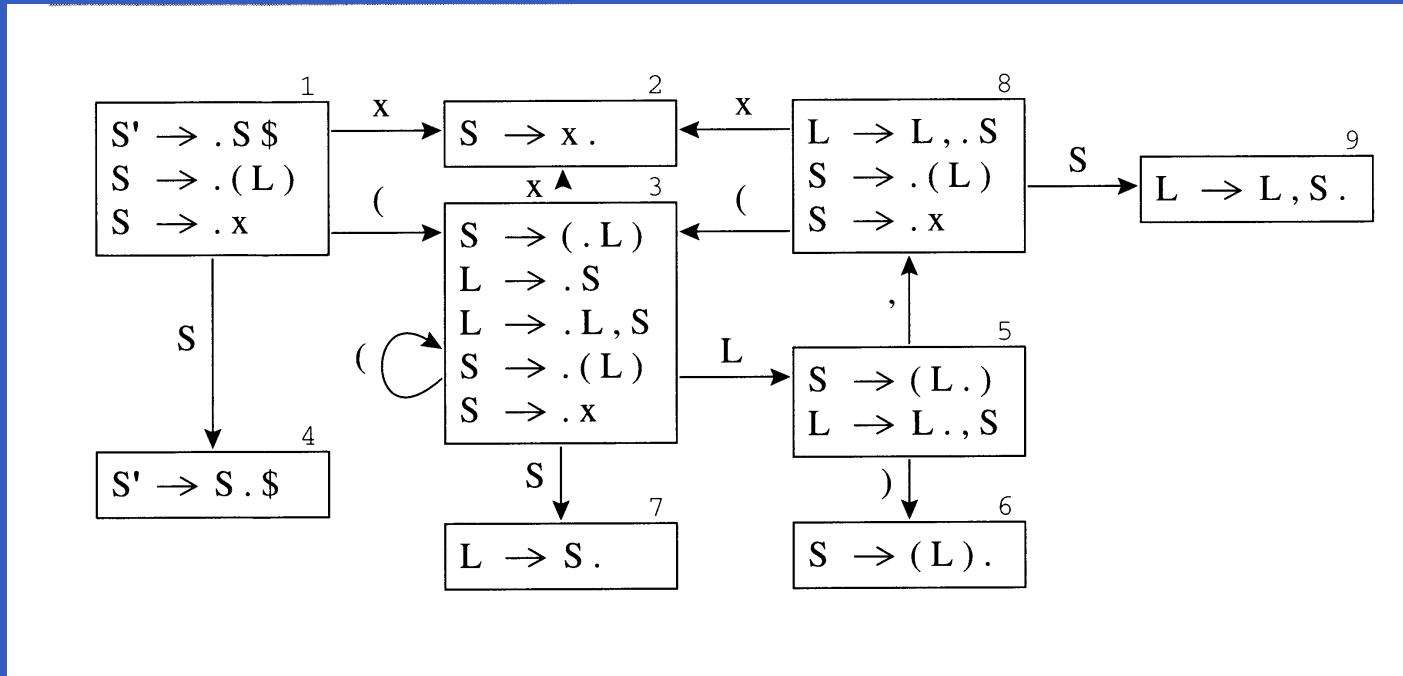
Exemple : reconnaître (x)



Config.: $\langle 1(3S7,)\$, 2 \rangle$

Action: Reduce 3

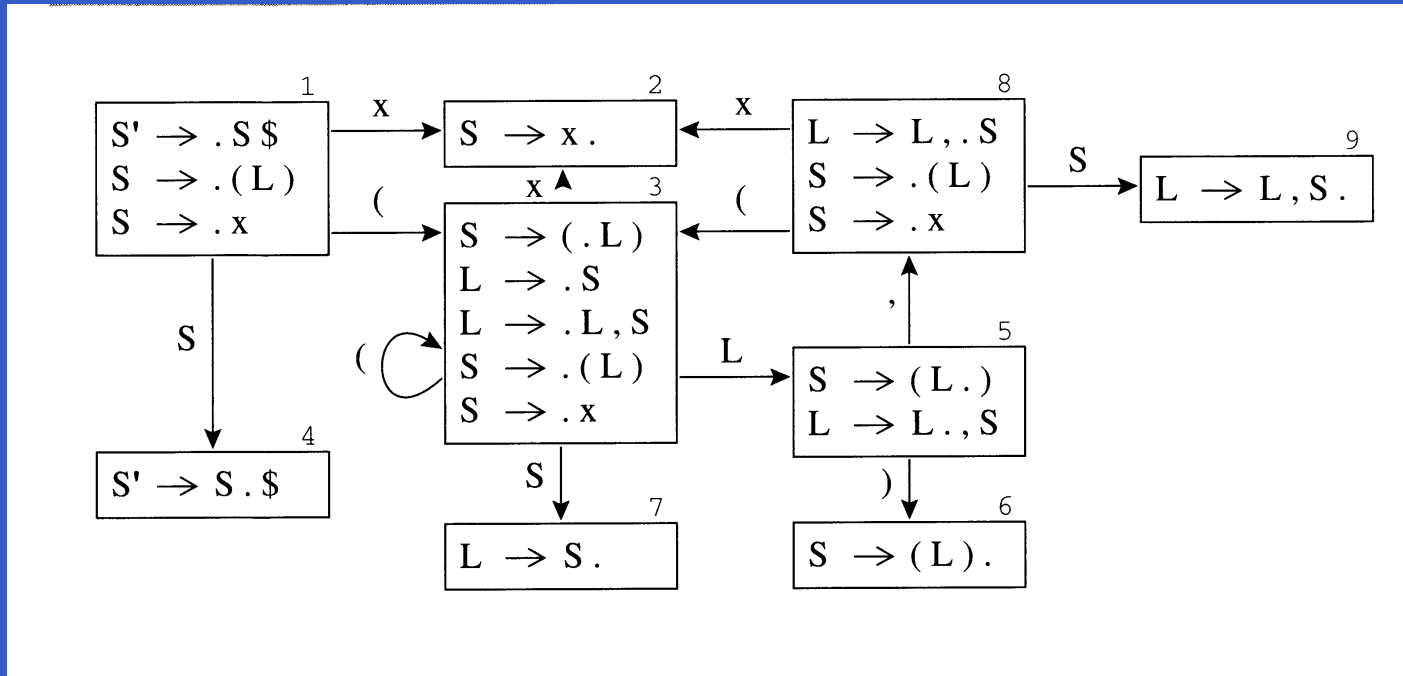
Exemple : reconnaître (x)



Config.: $\langle 1(3L5,)\$, 23 \rangle$

Action: Shift

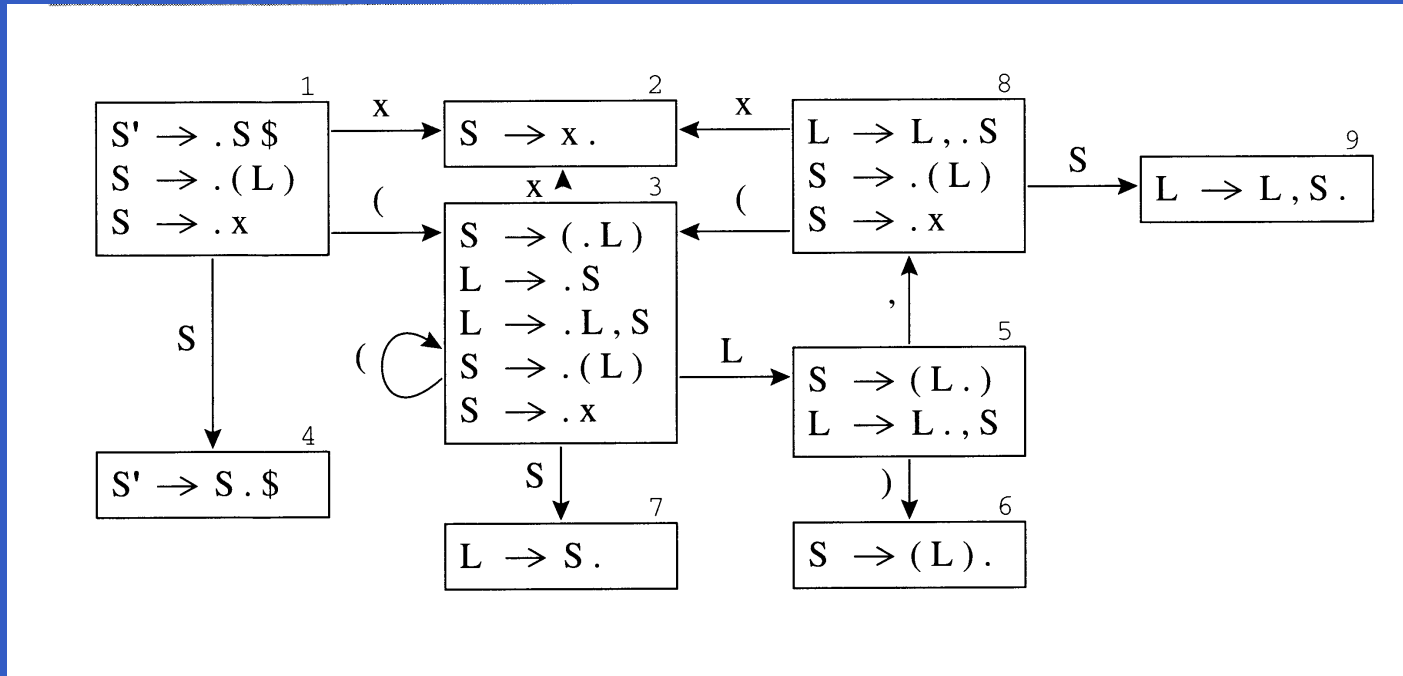
Exemple : reconnaître (x)



Config.: $\langle 1(3L5)6,)\$, 23 \rangle$

Action: Reduce 1

Exemple : reconnaître (x)



Config.: $\langle 1S4, \$, 231 \rangle$

Action: Accept

Exercice 2

- Simulez le fonctionnement de l'analyseur que vous avez construit à l'exercice 1 sur la chaîne aeyz \bar{z} d.

Exercice 2 – Solution

Pile	Entrée	Actions	Sortie
0	aeyzzd\$	Shift	
0a2	eyzzd\$	Shift	
0a2e3	yzzd\$	Shift	
0a2e3y5	zzd\$	Reduce 8	
0a2e3D4	zzd\$	Reduce 4	8
0a2C8	zzd\$	Shift	8,4
0a2C8z7	zd\$	Reduce 7	8,4
0a2C8F10	zd\$	Reduce 6	8,4,7
⋮	⋮	⋮	⋮

Exercice 1 – Solution

Pile	Entrée	Actions	Sortie
0a2C8	zd\$	Shift	8,4,7,6
0a2C8z7	d\$	Reduce 7	8,4,7,6
0a2C8F10	d\$	Reduce 6	8,4,7,6,7
0a2C8	d\$	Shift	8,4,7,6,7,6
0a2C8d15	\$	Reduce 1	8,4,7,6,7,6
0S12	\$	Accept	8,4,7,6,7,6,1

LR(k) : Introduction

- Différence par rapport à LR(0) : on doit maintenant tenir compte de la **prévision** !
- Par exemple :
 - Considérons un état du CFSM qui contient $A \rightarrow \alpha_1 \bullet \alpha_2$ et $B \rightarrow \gamma \bullet$
 - On a un **conflit *shift – reduce***
 - Si on n'a **pas** les caractères de $\text{First}^k(\alpha_2)$ sur l'entrée, il ne faut **pas essayer** le *shift*.
 - Mais dans quel **contexte** peut-on choisir à coup sûr ?

LR(k) : Introduction – 2

- Il faut retenir un **contexte** !
- Les *items* des états du CFSM seront maintenant de la forme : $[A \rightarrow \alpha_1 \bullet \alpha_2, u]$;
 - u indique le **contexte** : c'est l'ensemble de chaînes de k caractères qui peuvent suivre les productions de $A \rightarrow \alpha_1 \alpha_2$
 - On commence avec $[S' \rightarrow \bullet S \$, \varepsilon]$
- On adapte donc les algorithmes, la table des actions...

Analyseur LR(k) – CFSM – 1

Fermeture(I) **begin**

répéter

 | $I' \leftarrow I$;

 | **pour chaque** *item* $[A \rightarrow \alpha \bullet B\beta, \sigma] \in I, B \rightarrow \gamma \in G'$ **faire**

 | | **pour chaque** $u \in First^k(\beta\sigma)$ **faire** $I \leftarrow I \cup [B \rightarrow \bullet\gamma, u]$;

 | **jusqu'à** $I' = I$;

 | **return**(I) ;

end

Transition(I, X) **begin**

 | **return**(Fermeture($\{[A \rightarrow \alpha X \bullet \beta, u] \mid [A \rightarrow \alpha \bullet X\beta, u] \in$

 | $I\}\}$)) ;

end

Analyseur LR(k) – CFSM – 2

On construit la tables des actions comme suit :

pour chaque état s du CFSM faire

si s contient $[A \rightarrow \alpha \bullet a\beta, u]$ alors

pour chaque $u \in First^k(a\beta u)$ faire

└ Action $[s, u] \leftarrow Action[s, u] \cup Shift$;

sinon si s contient $[A \rightarrow \alpha \bullet, u]$, qui est la règle i alors

└ Action $[s, u] \leftarrow Action[s, u] \cup Reduce_i$;

sinon si s contient $[S' \rightarrow S\$ \bullet, \varepsilon]$ alors

└ Action $[s, \cdot] \leftarrow Action[s, \cdot] \cup Accept$;

Exercice 3

Construisez l'analyseur LR(1) pour la grammaire :

$$(1) \quad S' \rightarrow S\$$$

$$(2) \quad S \rightarrow A$$

$$(3) \quad A \rightarrow bB$$

$$(4) \quad A \rightarrow a$$

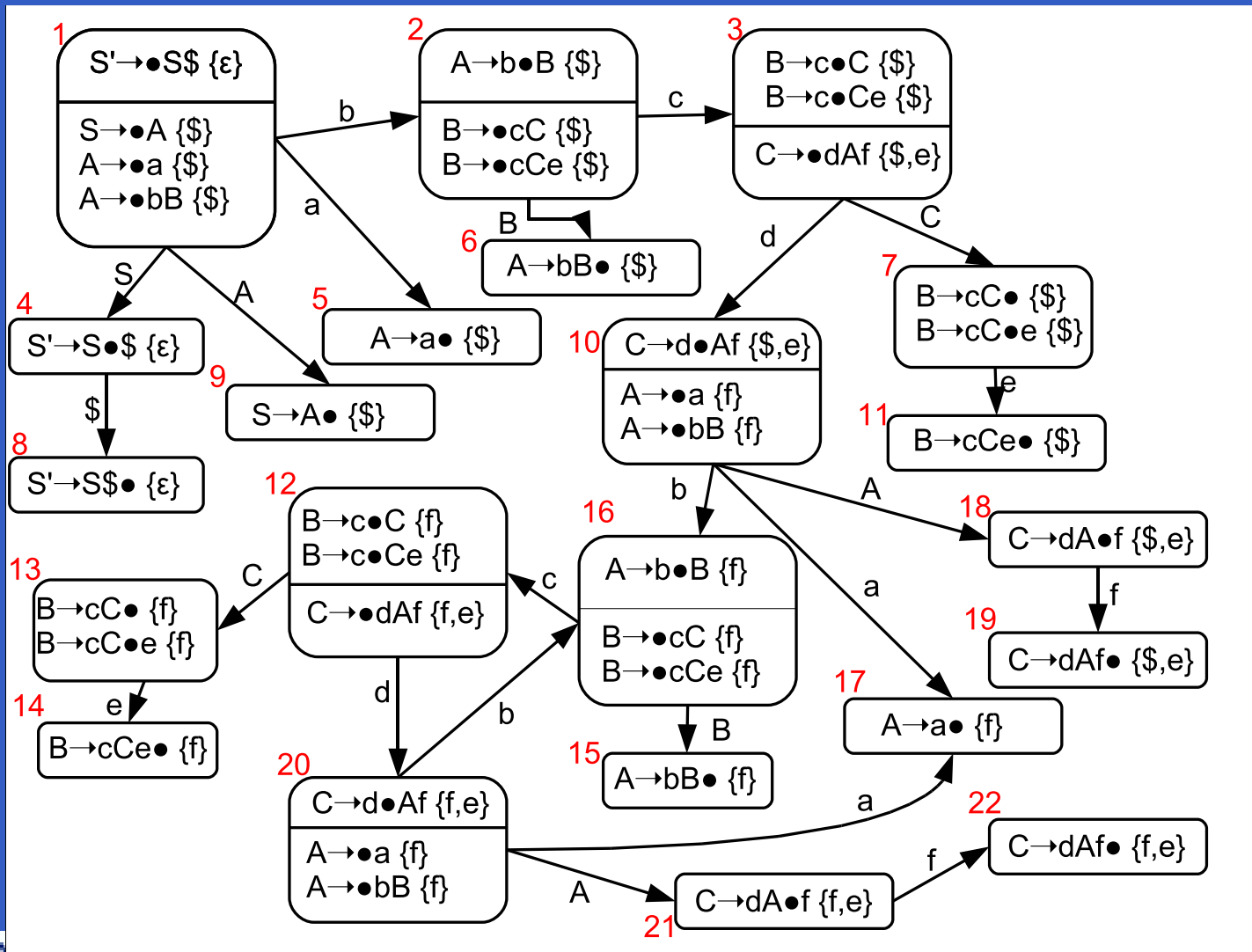
$$(5) \quad B \rightarrow cC$$

$$(6) \quad B \rightarrow cCe$$

$$(7) \quad C \rightarrow dAf$$

Cette grammaire est-elle LR(0) ? Justifiez.

Exercice 3 – Correction



Exercice 3 – Correction

	a	b	c	d	e	f	\$
1	S	S					
2			S				
3				S			
4							S
5							R4
6							R3
7					S		R5
8	Accept						

Exercice 3 – Correction

	a	b	c	d	e	f	\$
9							R2
10	S	S					
11							R6
12				S			
13					S	R5	
14						R6	
15						R3	
16			S				

Exercice 3 – Correction

	a	b	c	d	e	f	\$
17						R2	
18						S	
19					R7	R7	
20	S	S					
21						S	
22					R7	R7	

Exercice 4

Construisez l'analyseur LR(1) pour la grammaire :

$$(1) \quad S' \rightarrow S\$$$

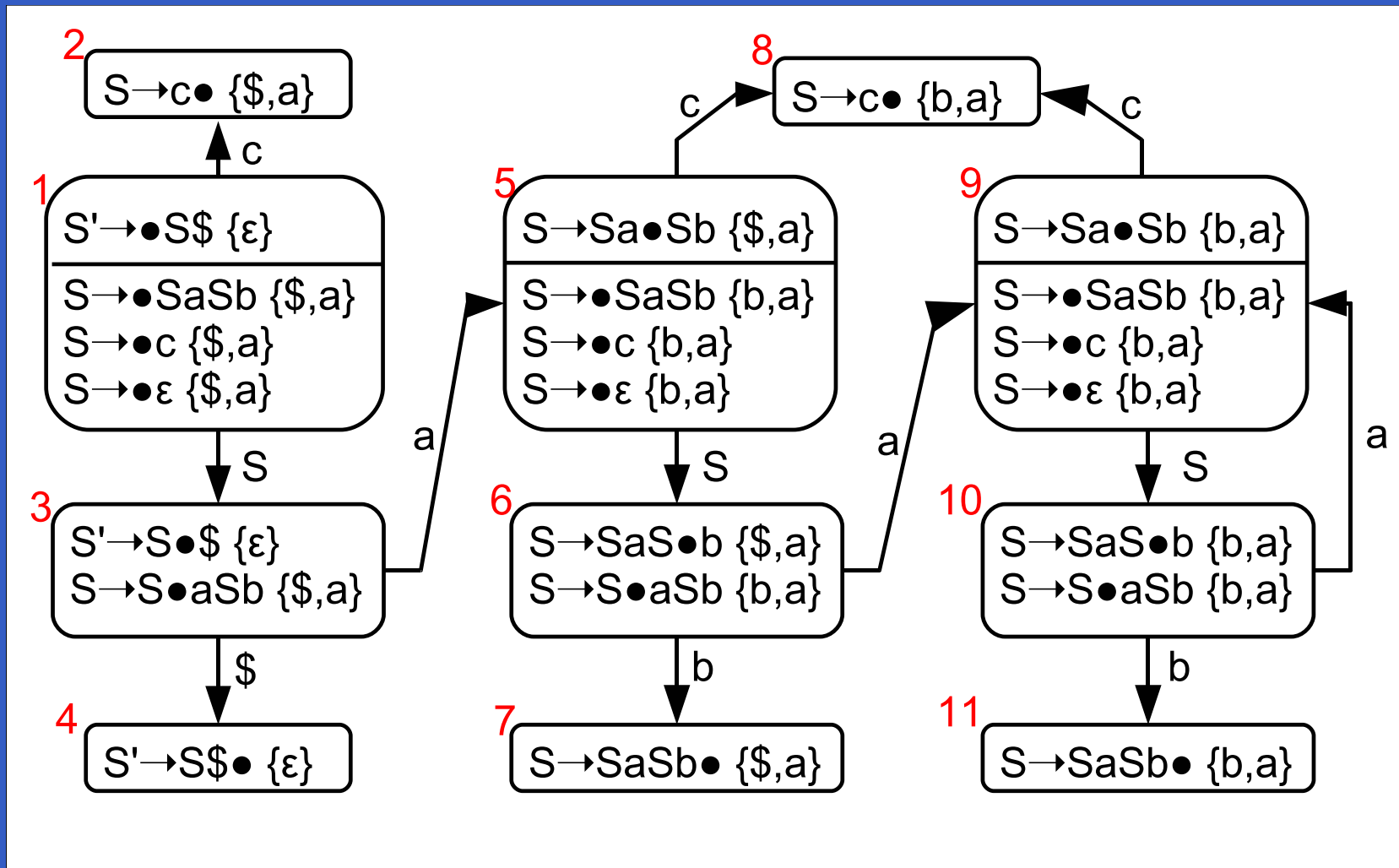
$$(2) \quad S \rightarrow SaSb$$

$$(3) \quad S \rightarrow c$$

$$(4) \quad S \rightarrow \varepsilon$$

Simulez-en le fonctionnement sur abacb.

Exercice 4 – Correction



Exercice 4 – Correction

	a	b	c	\$
1	R4		S	R4
2	R3			R3
3	S			S
4	Accept			
5	R4	R4	S	
6	S	S		

Exercice 4 – Correction

	a	b	c	\$
7	R2			R2
8	R3	R3		
9	R4	R4	S	
10	S	S		
11	R2	R2		

Exercice 4 – Correction

Stack	Input	Action	Output
1	abacb\$	Reduce 4	
1S3	abacb\$	Shift	4
1S3a5	bacb\$	Reduce 4	4
1S3a5S6	bacb\$	Shift	4,4
1S3a5S6b7	acb\$	Reduce 2	4,4
1S3	acb\$	Shift	4,4,2
1S3a5	cb\$	Shift	4,4,2

Exercice 4 – Correction

Stack	Input	Action	Output
1S3a5c8	b\$	Reduce 3	4,4,2
1S3a5S6	b\$	Shift	4,4,2,3
1S3a5S6b7	\$	Reduce 2	4,4,2,3
1S3	\$	Shift	4,4,2,3,2
1S3\$4		Accept	4,4,2,3,2