Traduction et Sémantique
Compléments sur AA
Traduction dirigée par la syntaxe

Luigi Santocanale
LIF, Université de Provence
Marseille, FRANCE

5 mars 2010

Plan

Analyse ascendante : compléments
- Analyse LR(1) . . .
  - . . . et LALR(1)
- En pratique

Traduction dirigée par la syntaxe
- Grammaires attribués

Plan

Analyse ascendante : compléments
- Analyse LR(1) ...
  - ... et LALR(1)
- En pratique

Traduction dirigée par la syntaxe
- Grammaires attribués

Prévision 1

- On considère un ensemble T de terminaux –
  symboles de prévision autorisant à réduire.
- Un item(1) :

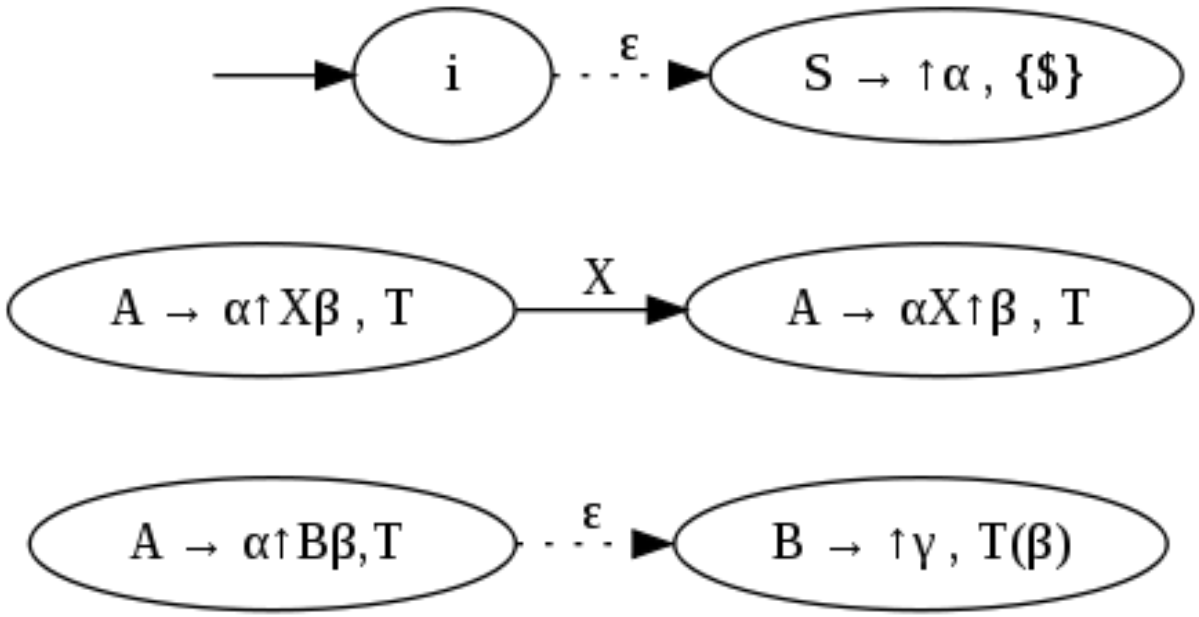
$$(A \to \alpha \uparrow \beta, T)$$

  où $T \subseteq \Sigma \cup \{\$\}$.
- Un item(1) $(A \to \alpha \uparrow \beta, T)$ est valide pour γ si

$$S \Rightarrow_D^* \delta A a w \Rightarrow_D \delta \alpha \beta a w$$
$$\gamma = \delta \alpha$$
$$a \in T\,.$$

Le AFN

Transitions :



où

$$T(\beta) = \begin{cases} First(\beta) \cup T & \text{si } Null(\beta) \\ First(\beta) & \text{sinon.} \end{cases}$$

Un exemple

$$
\begin{aligned}
S &\rightarrow A \\
A &\rightarrow B\,A \mid \varepsilon \\
B &\rightarrow a\,B \mid b
\end{aligned}
$$

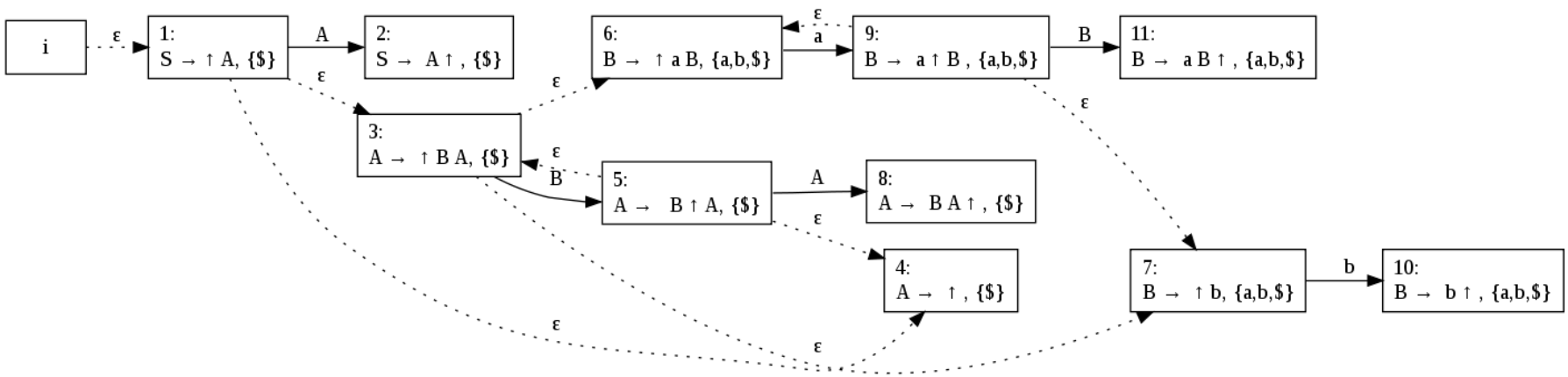
	nulll	First
S	oui	{a,b}
A	oui	{a,b}
B	non	{a,b}

On a

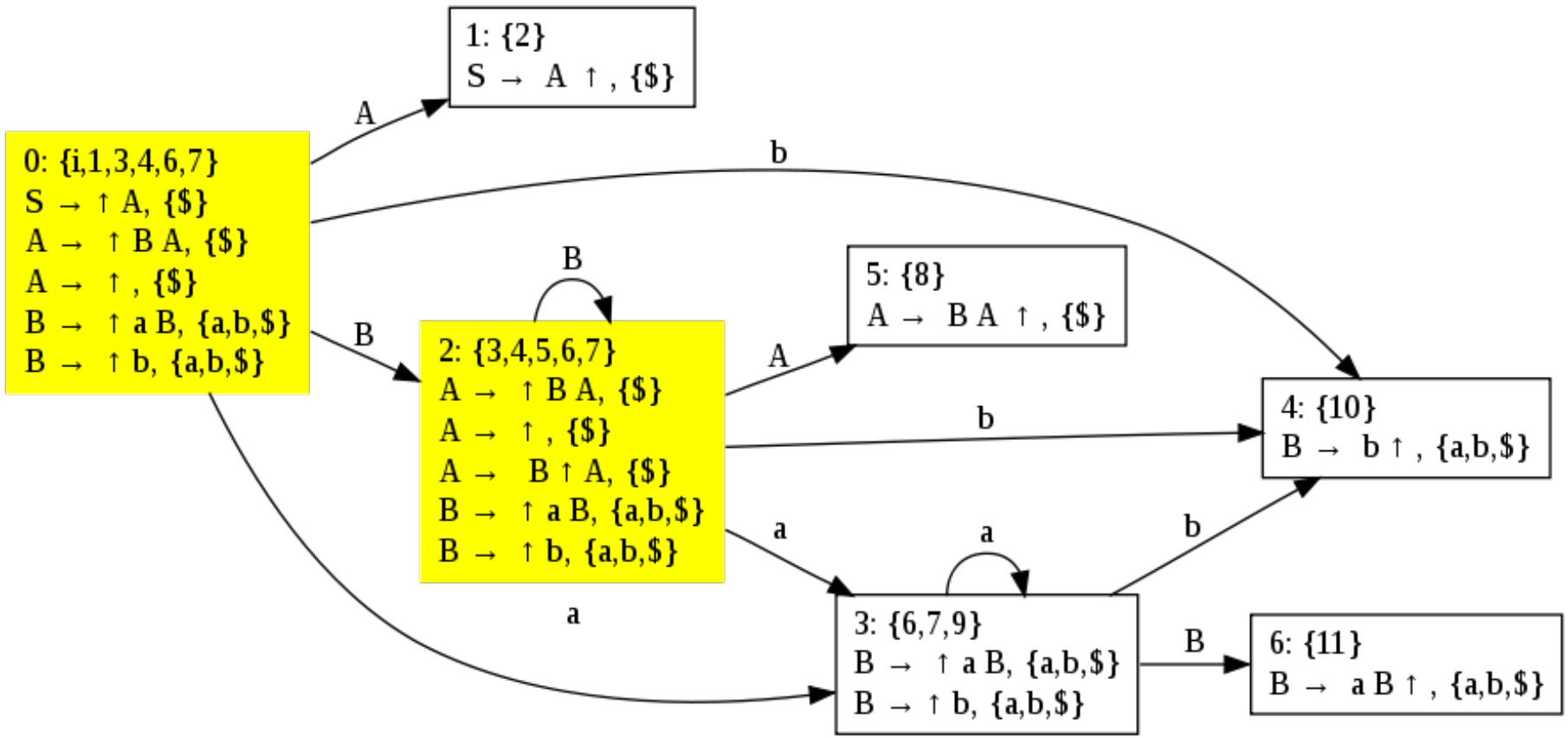
$$(A \rightarrow\uparrow BA, \{\$\}) \xrightarrow{\varepsilon} (B \rightarrow\uparrow aB, \{a, b, \$\})$$

car $A \in NULL$ et $First(A) = \{a, b\}$.

Le AFN


i
ε
1:
S → ↑ A, {$}
A
2:
S → A ↑ , {$}
ε
3:
A → ↑ B A, {$}
ε
B
5:
A → B ↑ A, {$}
A
8:
A → B A ↑ , {$}
ε
4:
A → ↑ , {$}
ε
ε
6:
B → ↑ a B, {a,b,$}
ε
a
9:
B → a ↑ B , {a,b,$}
B
11:
B → a B ↑ , {a,b,$}
ε
7:
B → ↑ b, {a,b,$}
b
10:
B → b ↑ , {a,b,$}

Le AFD


0: {i,1,3,4,6,7}
S → ↑ A, {$}
A → ↑ B A, {$}
A → ↑ , {$}
B → ↑ a B, {a,b,$}
B → ↑ b, {a,b,$}
1: {2}
S → A ↑ , {$}
2: {3,4,5,6,7}
A → ↑ B A, {$}
A → ↑ , {$}
A → B ↑ A, {$}
B → ↑ a B, {a,b,$}
B → ↑ b, {a,b,$}
3: {6,7,9}
B → ↑ a B, {a,b,$}
B → ↑ b, {a,b,$}
4: {10}
B → b ↑ , {a,b,$}
5: {8}
A → B A ↑ , {$}
6: {11}
B → a B ↑ , {a,b,$}
A
B
b
B
A
b
a
a
b
a
B

Un calcul

Pile	Entrée	Action	
0	*aabb*\$	*shift*	
0*a*3	*abb*\$	*shift*	
0*a*3*a*3	*bb*\$	*shift*	
0*a*3*a*3*b*4	*b*\$	*reduce*	$B \rightarrow b$
0*a*3*a*3*B*6	*b*\$	*reduce*	$B \rightarrow aA$
0*a*3*B*6	*b*\$	*reduce*	$B \rightarrow aA$
0*B*2	*b*\$	*shift*	
0*B*2*b*4	\$	*reduce*	$B \rightarrow b$
0*B*2*B*2	\$	*reduce*	$A \rightarrow \varepsilon$
0*B*2*B*2*A*5	\$	*reduce*	$A \rightarrow BA$
0*B*2*A*5	\$	*reduce*	$A \rightarrow BA$
0*A*1	\$	*reduce*	$S \rightarrow A$
0*S*	\$	*accepte*	

De LR(1) à LALR(1)

LA : LookAhead.

- Les AFD des items(1) sont très larges.
- On peut identifier les macro-états
  ayant exactement les mêmes items(0).
- En ce faisant, on peut introduire seulement
  des conflits réduire/réduire.
- Les outils comme YACC utilisent les AFD LALR(1).

De LR(1) à LALR(1)

LA : LookAhead.

- Les AFD des items(1) sont très larges.
- On peut identifier les macro-états
  ayant exactement les mêmes items(0).
- En ce faisant, on peut introduire seulement
  des conflits réduire/réduire.
- Les outils comme YACC utilisent les AFD ainsi réduits.

De LR(1) à LALR(1)

LA : LookAhead.

- Les AFD des items(1) sont très larges.
- On peut identifier les macro-états
  ayant exactement les mêmes items(0).
- En ce faisant, on peut introduire seulement
  des conflits réduire/réduire.
- Les outils comme YACC utilisent les AFD ainsi réduits.

De LR(1) à LALR(1)

LA : LookAhead.

- Les AFD des items(1) sont très larges.
- On peut identifier les macro-états
  ayant exactement les mêmes items(0).
- En ce faisant, on peut introduire seulement
  des conflits réduire/réduire.
- Les outils comme YACC utilisent les AFD ainsi réduits.

De LR(1) à LALR(1)

LA : LookAhead.

- Les AFD des items(1) sont très larges.
- On peut identifier les macro-états
  ayant exactement les mêmes items(0).
- En ce faisant, on peut introduire seulement
  des conflits réduire/réduire.
- Les outils comme YACC utilisent les AFD ainsi réduits.

Un exemple

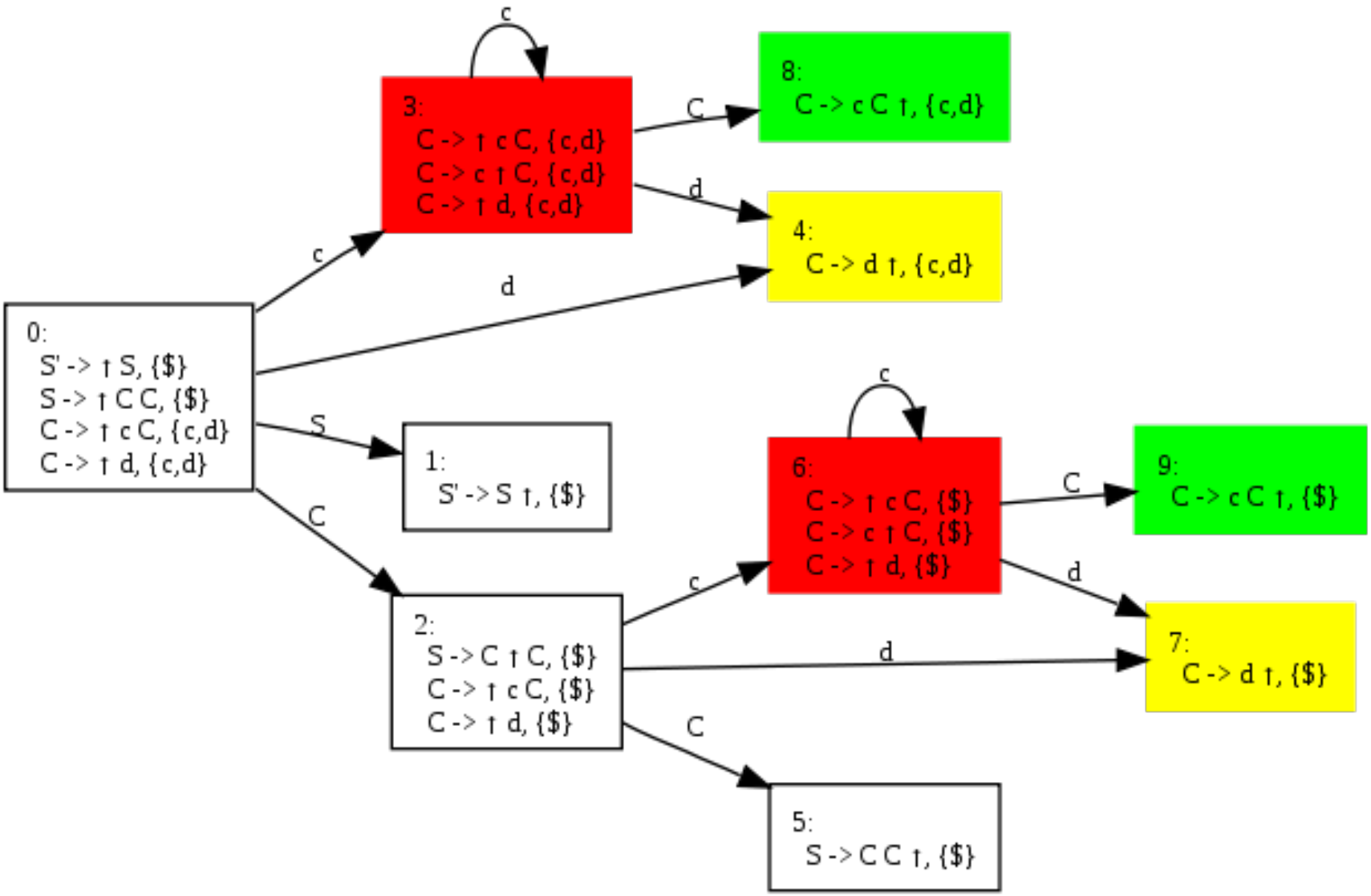
Soit $\mathscr{G}$:

$$\begin{aligned} S' &\to S \\ S &\to C\,C \\ C &\to c\,C \mid d \end{aligned}$$

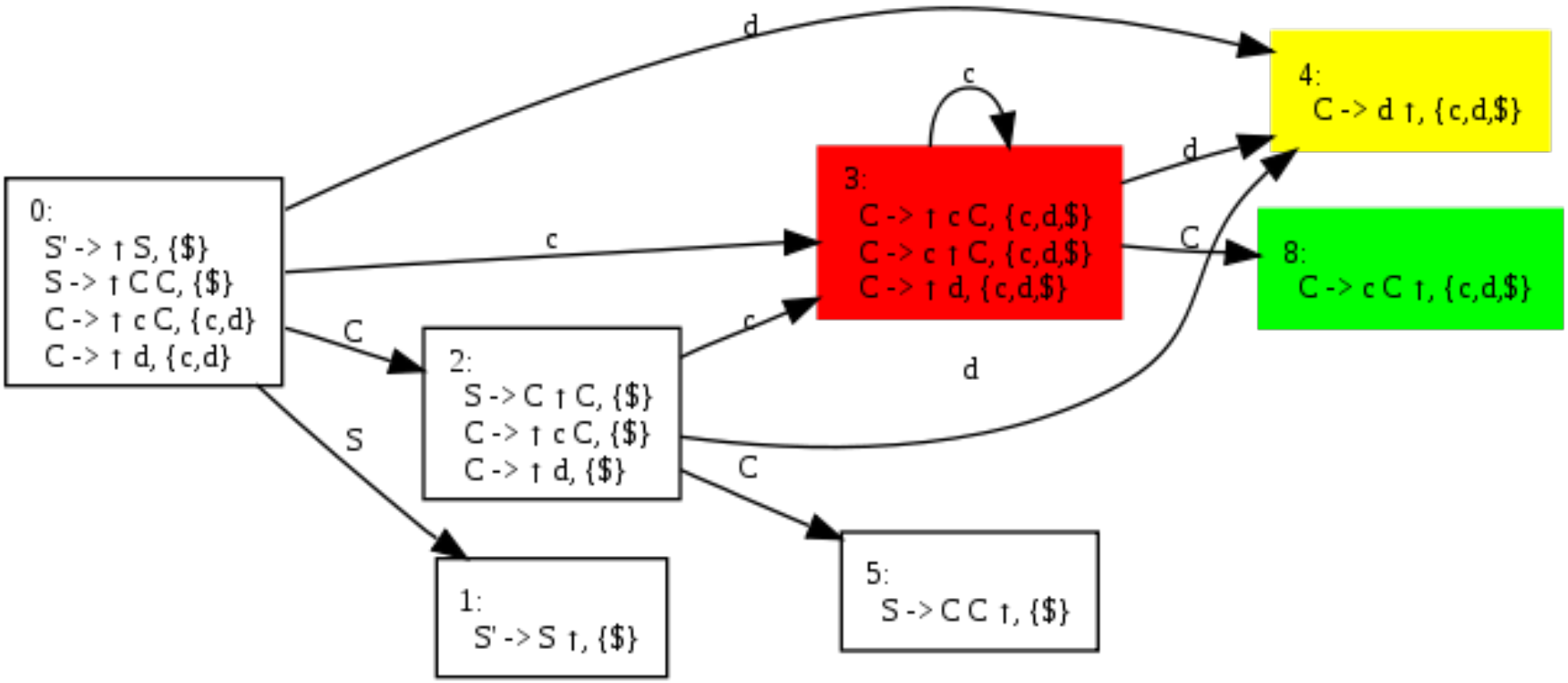
On a :

$$L(\mathscr{G}) = L(c^* d c^* d)\,.$$

Le AFD LR(1)


0:
S' -> ↑ S, {$}
S -> ↑ C C, {$}
C -> ↑ c C, {c,d}
C -> ↑ d, {c,d}
1:
S' -> S ↑, {$}
2:
S -> C ↑ C, {$}
C -> ↑ c C, {$}
C -> ↑ d, {$}
3:
C -> ↑ c C, {c,d}
C -> c ↑ C, {c,d}
C -> ↑ d, {c,d}
4:
C -> d ↑, {c,d}
5:
S -> C C ↑, {$}
6:
C -> ↑ c C, {$}
C -> c ↑ C, {$}
C -> ↑ d, {$}
7:
C -> d ↑, {$}
8:
C -> c C ↑, {c,d}
9:
C -> c C ↑, {$}
c
d
S
C

Le AFD LALR(1)


0:
S' -> ↑ S, {$}
S -> ↑ C C, {$}
C -> ↑ c C, {c,d}
C -> ↑ d, {c,d}
1:
S' -> S ↑, {$}
2:
S -> C ↑ C, {$}
C -> ↑ c C, {$}
C -> ↑ d, {$}
3:
C -> ↑ c C, {c,d,$}
C -> c ↑ C, {c,d,$}
C -> ↑ d, {c,d,$}
4:
C -> d ↑, {c,d,$}
5:
S -> C C ↑, {$}
8:
C -> c C ↑, {c,d,$}
d
c
C
S

Deux calculs

LR(1) :

Pile	Entrée	Action
0	*cd*\$	*shift*
0*c*3	*d*\$	*shift*
0*c*3*d*4	\$	*error*

LALR(1) :

Pile	Entrée	Action	
0	*cd*\$	*shift*	
0*c*3	*d*\$	*shift*	
0*c*3*d*4	\$	*reduce*	$C \longrightarrow c$
0*c*3*C*8	\$	*reduce*	$C \longrightarrow cC$
0*C*2	\$	*error*	

Défs et remarques

- Un grammaire est LR(1) si
  l'AFD des items(1) ne présente pas de conflits.
- Un grammaire est LALR(1) si l'AFD LALR(1)
  – c'est-à-dire l'AFD des items(1) après reduction –
  ne présente pas de conflits.
- Pour construire l'AFD LALR(1)
  n'est pas necessaire construire l'AFD LR(1).

À l'origine des conflits

- La grammaire est ambiguë :
  - grammairs d'expressions :

```
x - y - z
```

  - précédence des operateurs :

```
? x - y
```

  - if-then(-else) en suspens :

```
if true then if false then print a else print b
```

- Prévision limitée.
- Recherche des conflits : exercice en TD.

Prévision limitée : un exemple

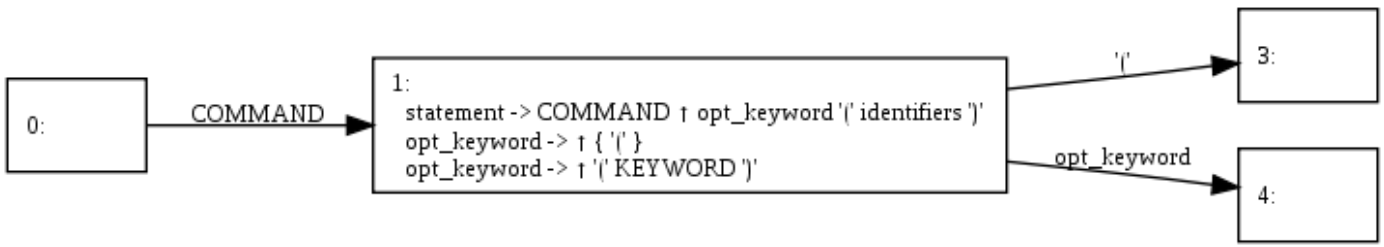
```
%token COMMAND KEYWORD IDENTIFIER
%%
statement: COMMAND opt_keyword '(' identifiers ')'
;

opt_keyword: /* vide */
   | '(' KEYWORD ')'
;

identifiers : /* vide */
   | identifiers IDENTIFIER
;
%%

>> byacc -g -v lla.y
byacc: 1 rule never reduced
byacc: 1 shift/reduce conflict.
```

Les outils : `name.dot` et `name.output`



Plan

Analyse ascendante : compléments
Analyse LR(1) . . .
. . . et LALR(1)
En pratique

Traduction dirigée par la syntaxe
Grammaires attribués

Définitions dirigées par la syntaxe

Grammaire attribuée :
toute production possède un ensemble de règles sémantiques :

Production	Règle sémantique
⋮	
$A \rightarrow X_1 \dots X_n$	a_1
	a_2
	⋮
	a_k
⋮	

Chaque règle sémantique (ou action) a_i :

- calcule un attribut d'un symbole parmi $A, X_1, \dots, X_n$, ou
- produit un effet de bord.

Grammaire attribuées pour une calculette

Production	Règle sémantique
$L \rightarrow E \backslash \mathtt{n}$	$\mathit{Imprimer}(E.\mathit{val})$
$E \rightarrow E_1 + T$	$E.\mathit{val} := E_1.\mathit{val} + T.\mathit{val}$
$E \rightarrow T$	$E.\mathit{val} := T.\mathit{val}$
$T \rightarrow T_1 * F$	$T.\mathit{val} := T_1.\mathit{val} \cdot F.\mathit{val}$
$T \rightarrow F$	$T.\mathit{val} := F.\mathit{val}$
$F \rightarrow (E)$	$F.\mathit{val} := E.\mathit{val}$
$F \rightarrow \mathtt{chiffre}$	$F.\mathit{val} := \mathtt{chiffre}.\mathit{vallex}$