Linguaggi e Traduttori – Esercizi LR(1) e SLR(1)

Prof. Marco Gavanelli

10 giugno 2018

Esercizio 1

Si consideri la grammatica $G = \langle \{a, b, d\}, \{S, A, B\}, P, S \rangle$, dove:

$$P = \begin{array}{l} S \to Ab \mid aB \\ A \to \epsilon \mid Ba \\ B \to dSa \mid BAb \end{array}$$

Si dica se la grammatica è LR(1). Qualora la grammatica risulti LR(1), si mostri come l'automa riconosce la frase *adbab*.

Si dica se la grammatica è SLR(1). Qualora la grammatica risulti LR(1), si mostri come l'automa riconosce la frase *b*.

Si dica se la grammatica è LR(0).

Esercizio 2

Si consideri la grammatica $G = \langle \{a, b, (,)\}, \{S\}, P, S \rangle$, dove:

$$P = \quad S \to a \mid (S) \mid SbS$$

Si dica se la grammatica è LR(1).

Si dica se la grammatica è SLR(1).

Si dica se la grammatica è LR(0).

Esercizio 3

Si consideri la grammatica $G = \langle \{a, b, d, \}, \{S, A, B\}, P, S \rangle$, dove:

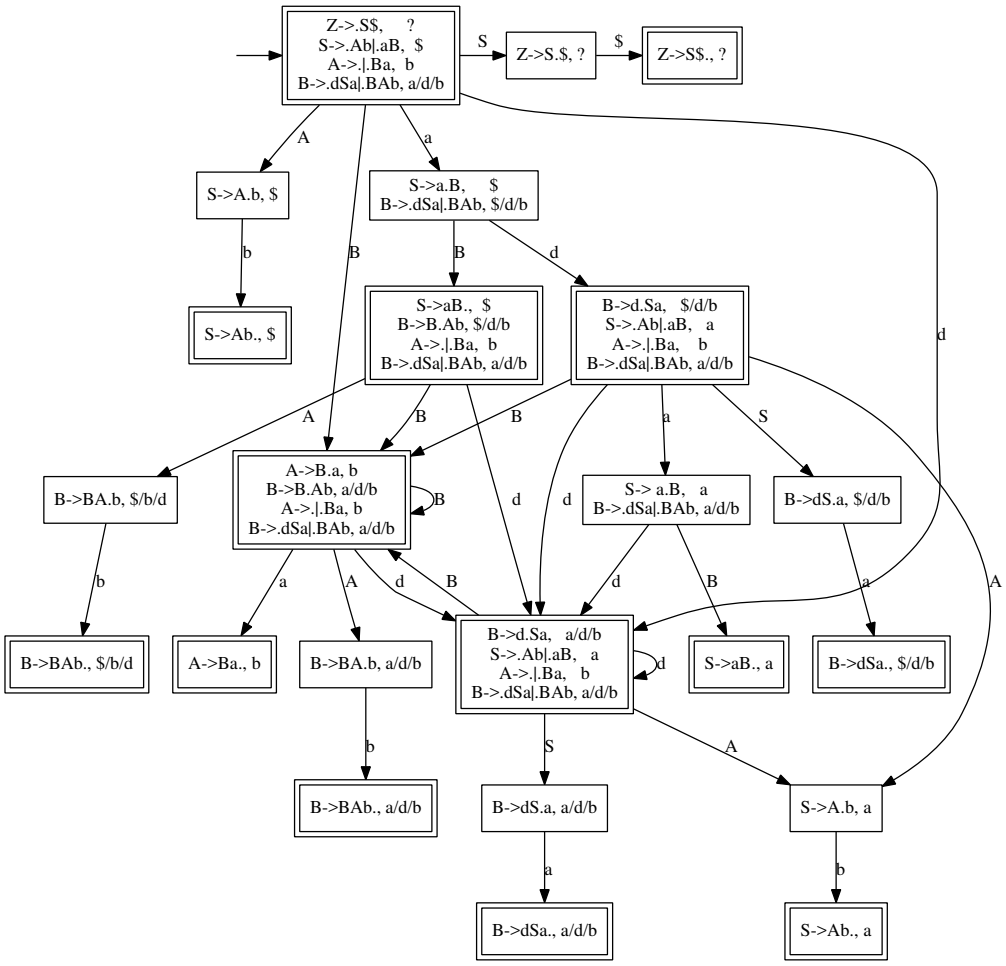
$$P = \begin{array}{l} S \to Ab \mid B \mid aBb \\ B \to A \\ A \to d \end{array}$$

- Si dica se la grammatica è SLR(1).
- Si dica se la grammatica è LR(0).
- Si dica se la grammatica è LR(1).

Qualora la grammatica risulti LR(0), SLR(1) o LR(1), si costruisca l'automa corrispondente. Si mostri poi il riconoscimento delle frasi *db*, *d* e *adb*.

Soluzione 1

Per verificare se la grammatica è LR(1), scriviamo l'automa LR(1):



L'automa non presenta conflitti, quindi la grammatica è LR(1).

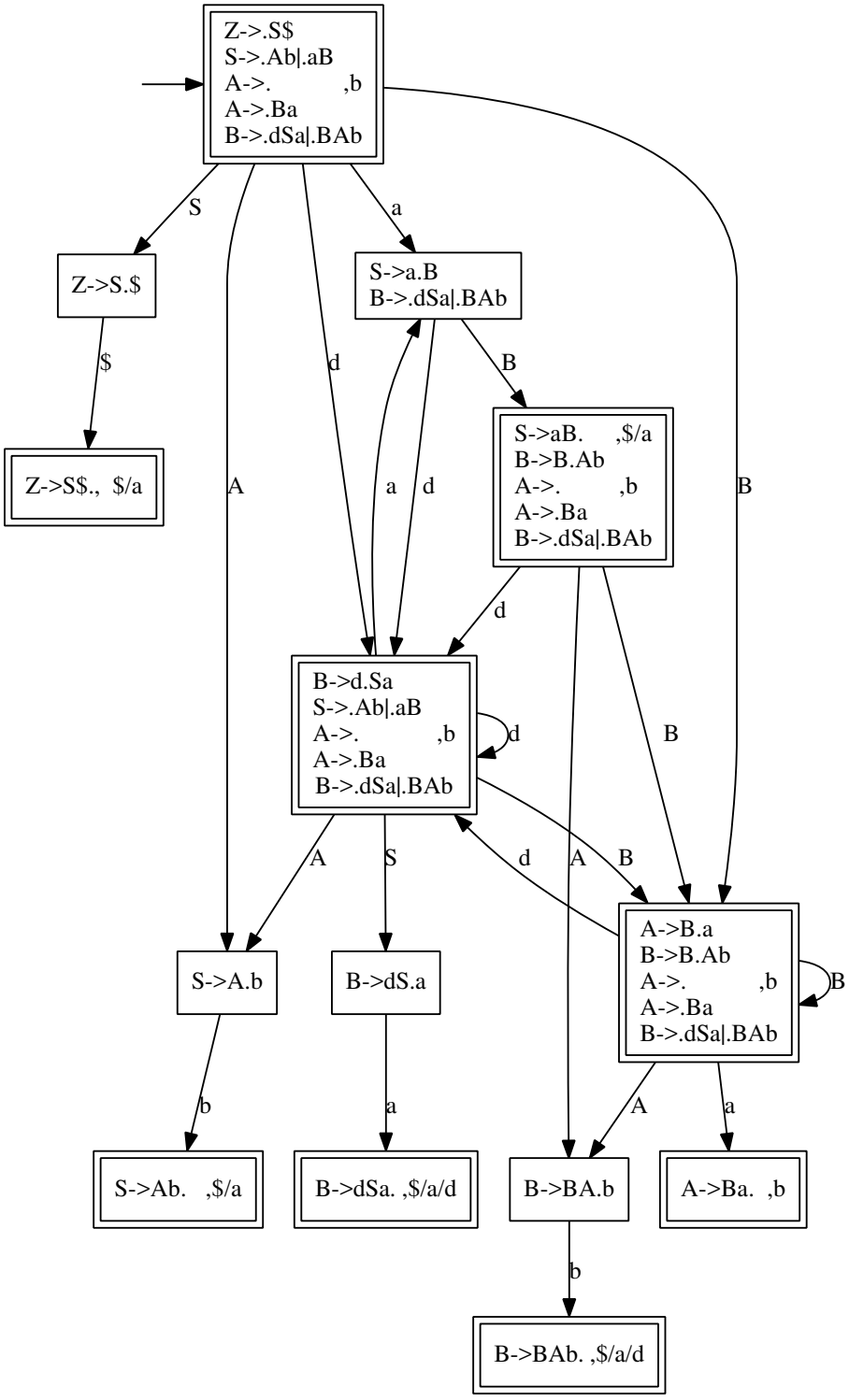
Stack	Input
	adbab$
a	dbab$
ad	bab$
adA	bab$
adAb	ab$
adS	ab$
adSa	b$
aB	b$
aBA	b$
aBAb	$
aB	$
S	$
S$	
Z	

Stringa riconosciuta.

Per verificare se la grammatica è SLR(1), calcoliamo l'insieme FOLLOW dei non terminali:

	$FIRST$	$FOLLOW$
S	$\{a, b, d\}$	$\{\$, a\}$
A	$\{\epsilon, d\}$	$\{b\}$
B	$\{d\}$	$\{\$, a, d\}$

Poi creiamo l'automa SLR(1), che è identico all'automa LR(0), con la differenza che in corrispondenza degli item di riduzione si riporta anche il FOLLOW del nonterminale a sinistra della freccia $\rightarrow$.


Z->.S$
S->.Ab|.aB
A->. ,b
A->.Ba
B->.dSa|.BAb
S
a
Z->S.$
S->a.B
B->.dSa|.BAb
$
d
B
Z->S$., $/a
A
a
d
S->aB. ,$/a
B->B.Ab
A->. ,b
A->.Ba
B->.dSa|.BAb
B
d
B->d.Sa
S->.Ab|.aB
A->. ,b
A->.Ba
B->.dSa|.BAb
d
B
A
S
d
A
B
S->A.b
B->dS.a
A->B.a
B->B.Ab
A->. ,b
A->.Ba
B->.dSa|.BAb
B
b
a
A
a
S->Ab. ,$/a
B->dSa. ,$/a/d
B->BA.b
A->Ba. ,b
b
B->BAb. ,$/a/d

L'automa non presenta conflitti, quindi la grammatica risulta SLR(1).

Stack	Input
	b$
A	b$
Ab	$
S	$
S$	
Z	

Stringa riconosciuta.

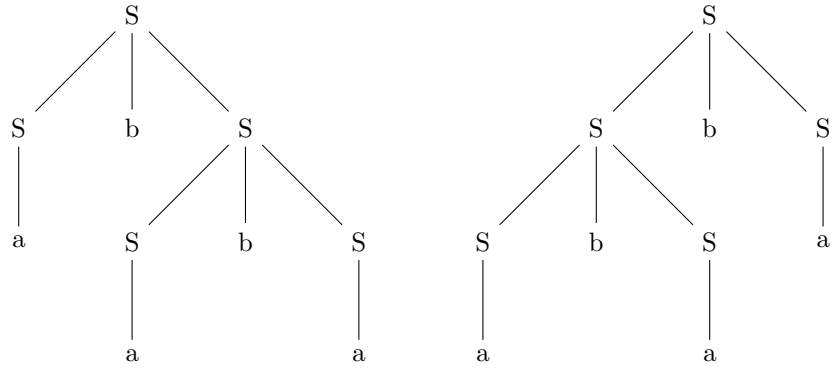
La grammatica non è LR(0): lo può notare anche osservando l'automa SLR(1). Infatti, si immagini l'automa SLR(1) senza i simboli tratti dall'insieme FOLLOW: già nel nodo iniziale si avrebbe un conflitto shift-reduce: in tale nodo sarebbe infatti applicabile sia la riduzione $A \to \epsilon$ sia un'operazione di shift, ad esempio di a.

Il conflitto scompare nell'automa LR(1) in quanto la riduzione $A \to \epsilon$ è applicabile solo se il carattere successivo è b; siccome nessuna shift è applicabile col carattere b, nell'automa LR(1) non è presente il conflitto.

Nell'automa LR(0) sarebbe presente anche un conflitto reduce-reduce, in quanto è presente un nodo con due item di riduzione: $S \to aB.$ e $A \to .$ Se non si guarda il carattere successivo nell'input, è impossibile stabilire quale delle due riduzioni va applicata in quel nodo.

Soluzione 2

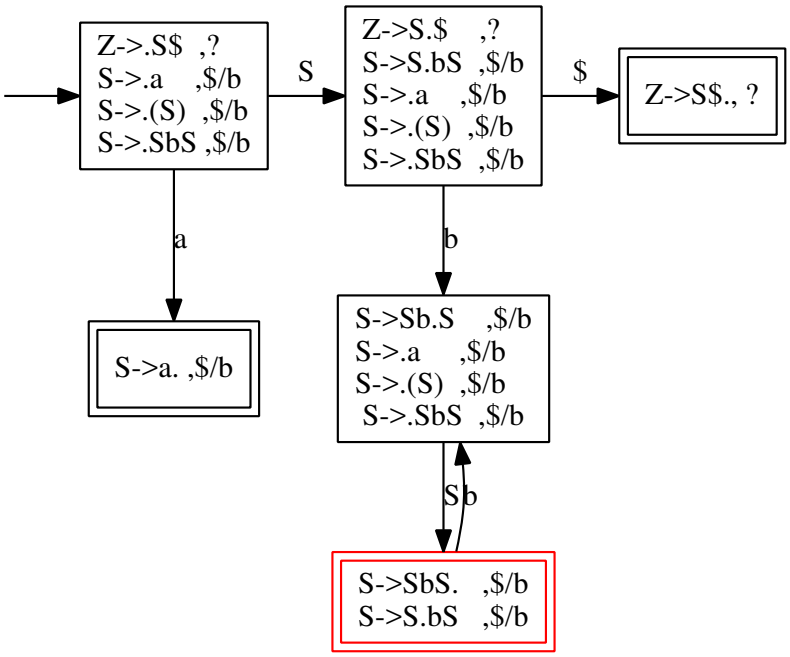
La grammatica data è ambigua, come si può vedere dal fatto che la frase *ababa* ha due alberi di derivazione diversi:



In sostanza, non viene data l'associatività dell'operatore b.

Di conseguenza, la grammatica non può essere LR(1), né tanto meno SLR(1) o LR(0).

Se non ci si accorge dell'ambiguità, si può generare l'automa LR(1):



a questo punto della costruzione dell'automa (l'automa non è ancora completo) si vede già che c'è un conflitto shift-reduce: nello stato indicato in rosso

- si effettua la riduzione $S \to SbS$ se il carattere successivo è \$ o b (come indicato dall'item $S \to SbS., \$/b$)
- e si effettua shift se il carattere successivo è b (come indicato dall'item $S \to S.bS, \$/b$)

Quindi l'automa non è in grado di decidere quale operazione effettuare quando il carattere successivo è b.

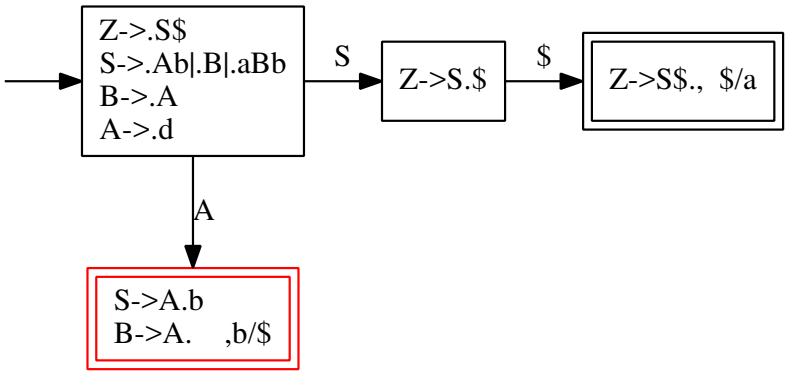
Soluzione 3

Per verificare se la grammatica è SLR(1) si costruisce l'automa LR(0), associando ad ogni item di riduzione del tipo $X \to \alpha.$ l'insieme FOLLOW(X).

Calcoliamo quindi l'insieme FOLLOW dei non terminali:

	$FIRST$	$FOLLOW$
S	$\{a, d\}$	$\{\$\}$
A	$\{d\}$	$\{b, \$\}$
B	$\{d\}$	$\{b, \$\}$

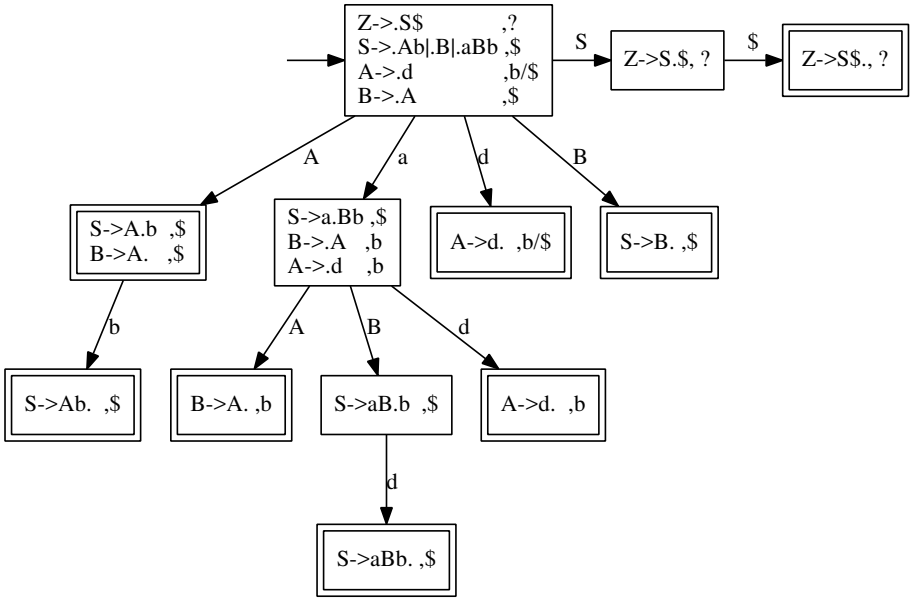
Cominciamo a scrivere l'automa:



Scrivendo l'automa, troviamo un conflitto nel nodo indicato in rosso (quindi non è necessario proseguire nella costruzione dell'automa). Qualora l'automa si trovi nello stato indicato in rosso e il prossimo simbolo di input sia b, l'automa non è in grado di decidere se effettuare *shift* b (dovuto all'item $S \to A.b$) oppure *reduce* $B \to A$ (dovuto all'item $B \to A., b$). Si è quindi in presenza di un conflitto shift-reduce.

Visto che la grammatica non è SLR(1), ovviamente non è neanche LR(0).

Per verificare se la grammatica è LR(1), scriviamo l'automa LR(1):



L'automa non presenta conflitti, quindi la grammatica è LR(1).
Riconoscimento stringhe LR(1):

Stack	Input
	db$
d	b$
A	b$
Ab	$
S	$
S$	
Z	

Stringa riconosciuta.

Stack	Input
	d$
d	$
A	$
B	$
S	$
S$	
Z	

Stringa riconosciuta.

Stack	Input
	adb$
a	db$
ad	b$
aA	b$
aB	b$
aBb	$
S	$
S$	
Z	

Stringa riconosciuta.