

1 Esercizio 1

1.1 Creazione dell'automa caratteristico

Creiamo l'automa caratteristico \mathcal{A} per il parsing bottom-up SLR(1). Iniziamo aggiungendo una nuova produzione con un nuovo non terminale che produce lo start symbol:

$$S' \rightarrow \cdot S$$

Procediamo con il calcolo della chiusura LR(0), che consiste nell'includere tutte le produzioni in cui il non terminale preceduto dal marker funge da driver, e ripetiamo questo processo in modo ricorsivo, ottenendo così il risultato finale:

$$S \rightarrow \cdot AaB \mid \cdot b$$

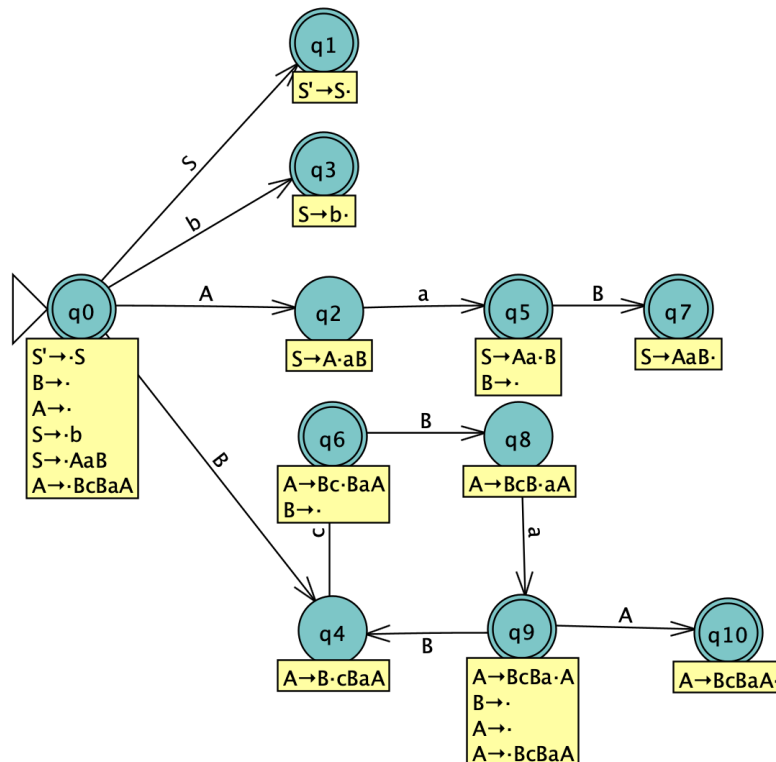
$$A \rightarrow \cdot BcBaA \mid \cdot$$

$$B \rightarrow \cdot$$

Questo sarà lo stato iniziale 0 dell'automa. Analizzando le transizioni da questo stato, otteniamo i seguenti kernel degli stati di arrivo:

- Transizione con S : il kernel è $\{S' \rightarrow S\cdot\}$, corrispondente allo stato 1. La chiusura di questo stato è vuota poiché il marker (\cdot) si trova alla fine della produzione, rendendo questo stato reducing (stato finale). Inoltre, questo stato è anche lo stato di accettazione.
- Transizione con A : il kernel è $\{S \rightarrow A\cdot aB\}$, corrispondente allo stato 2. Anche in questo caso, la chiusura è vuota perché il marker non precede alcun non terminale.
- Transizione con b : il kernel è $\{S \rightarrow b\cdot\}$, corrispondente allo stato 3. La chiusura è vuota e, essendo uno stato reducing, è anche uno stato finale.
- Transizione con B : il kernel è $\{A \rightarrow B\cdot cBaA\}$, corrispondente allo stato 4. La chiusura di questo stato è vuota.

Notiamo che lo stato 0 ha due item di riduzione $A \rightarrow \cdot$ e $B \rightarrow \cdot$, quindi è anche uno stato finale. Proseguendo analogamente alla creazione degli altri stati seguendo le transizioni da quelli appena calcolati otteniamo l'automa seguente:



1.2 Creazione della tabella di parsing SLR(1)

Calcoliamo i set di *first* e *follow*:

Non Terminale	First	Follow
S	$\{a, b, c\}$	$\{\$ \}$
A	$\{c, \epsilon\}$	$\{a\}$
B	$\{\epsilon\}$	$\{c, a, \$ \}$

Costruiamo ora la tabella di parsing. Le transizioni dei terminali sono *shift*, quelle dei non terminali sono *GOTO*, mentre le riduzioni avvengono nei *Follow* dei reducing items.

Reducing items:

- $R_1: S \rightarrow AaB$.
- $R_2: S \rightarrow b$.
- $R_3: A \rightarrow BcBaA$.
- $R_4: A \rightarrow \cdot$.
- $R_5: B \rightarrow \cdot$.

Stato	a	b	c	\$	A	B	S
0	R4 / R5	S3	R5	R5	2	4	1
1				acc			
2	S5						
3				R4			
4			S6				
5	R5		R5	R5		7	
6	R5		R5	R5		8	
7				R1			
8	S9						
9	R4 / R5		R5	R5	10	4	
10	R3						

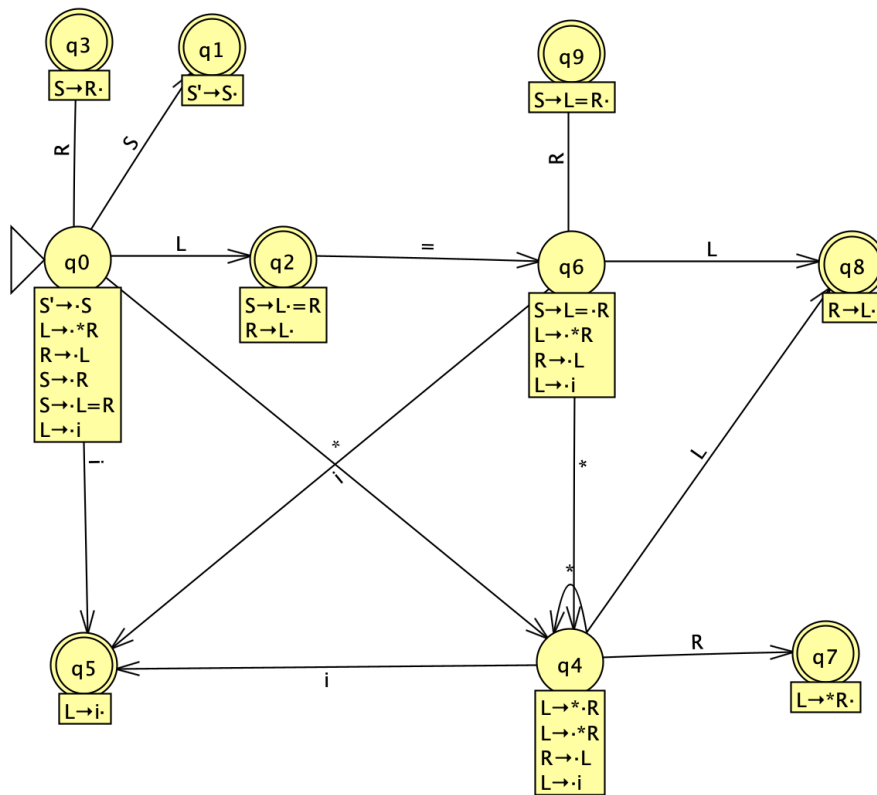
1.3 Risposte alle domande

1. Gli stati dell'automa \mathcal{A} sono 11.
2. Le mosse di shift sono 4.
3. Le mosse di reduce sono 17.
4. Sono presenti due conflitti evidenziati in giallo nella tabella.
5. I conflitti sono:
 - $T[I, a]$: conflitto *reduce/reduce*, produzioni coinvolte: $A \rightarrow \epsilon$ e $B \rightarrow \epsilon$.
 - $T[I[BcBa], a]$: conflitto *reduce/reduce*, produzioni coinvolte: $A \rightarrow \epsilon$ e $B \rightarrow \epsilon$.

2 Esercizio 2

2.1 Creazione dell'automa caratteristico

Lo svolgimento è analogo all'esercizio precedente:



2.2 Creazione della tabella di parsing SLR(1)

Calcoliamo i set di *first* e *follow*:

Non Terminale	First	Follow
S	$\{*, id\}$	$\{\$ \}$
L	$\{*, id\}$	$\{=, \$ \}$
R	$\{*, id\}$	$\{=, \$ \}$

Reducing items:

- $R_1: S \rightarrow L = R \cdot$
- $R_2: S \rightarrow R \cdot$
- $R_3: L \rightarrow *R \cdot$
- $R_4: L \rightarrow id \cdot$
- $R_5: R \rightarrow L \cdot$

Calcoliamo la tabella di parsing:

Stato	*	=	id	\$	L	R	S
0	S4		S5		2	3	1
1				acc			
2		S6 / R5		R5			
3				R2			
4	S4		S5		8	7	
5		R4		R4			
6	S4		S5		8	9	
7		R3		R3			
8		R5		R5			
9				R1			

2.3 Risposte alle domande

1. Gli stati dell'automa \mathcal{A} sono 10.
2. Le mosse di shift sono 7.
3. Le mosse di reduce sono 9.
4. E' presente un conflitto evidenziato in giallo nella tabella.
5. $T[I[L], =]$: conflitto *shift/reduce*, in particolare sono coinvolte l'azione di *shift 2* e la produzione $R \rightarrow L$

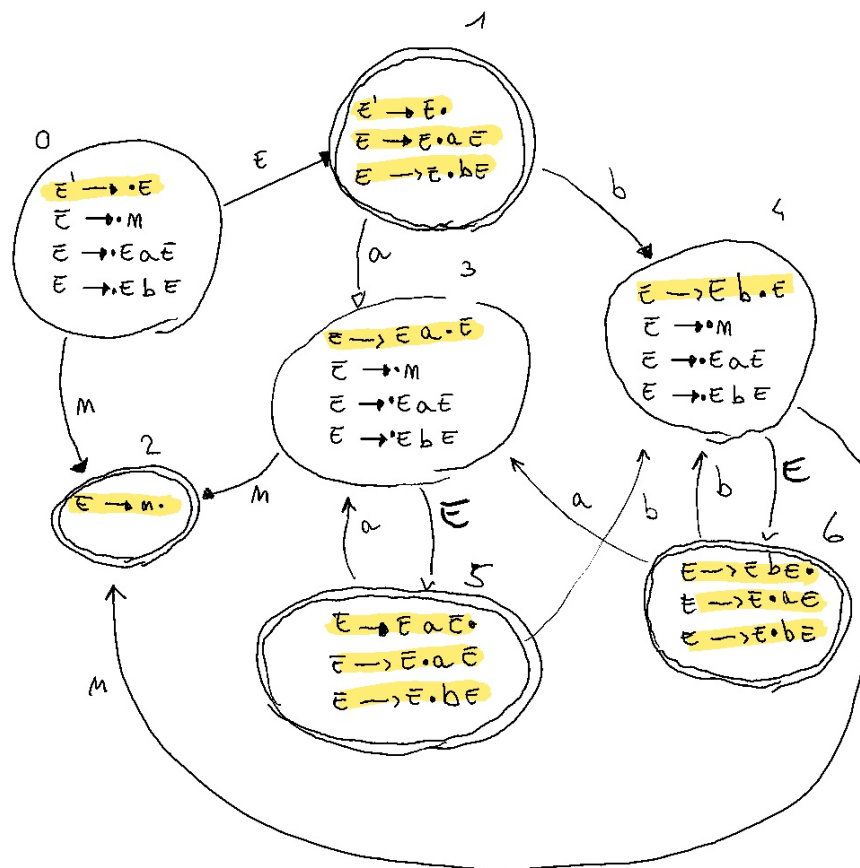
3 Esercizio 3

3.1 Soluzione

Analizziamo i conflitti della grammatica G_1 data:

- $[P[EaE], a]$: in questo stato, leggendo il simbolo terminale a , il parser può effettuare una riduzione con la produzione $E \rightarrow EaE$ oppure eseguire un'operazione di *shift* del terminale a . La decisione determina l'associatività dell'operatore a . Per esempio se il parser eseguisse l'azione di *reduce* garantirebbe l'associatività a sinistra. Questo conflitto può essere ignorato poiché non influisce sulla precedenza tra a e b .
- $[P[EbE], b]$: in modo analogo al caso precedente, ma relativo al terminale b , quindi possiamo ignorare anche questo conflitto.
- $[P[EaE], b]$: in questo stato, leggendo il terminale b , il parser può ridurre con $E \rightarrow EaE$ oppure eseguire uno *shift* del terminale b . Per assegnare una precedenza più alta all'operatore a rispetto a b , il parser deve eseguire la riduzione. In questo modo, l'espressione EaE viene risolta prima che b venga processato.
- $[P[EbE], a]$: in modo analogo al precedente per dare priorità all'operatore a rispetto a b , il parser dovrebbe scegliere l'azione di *shift*. In questo modo, l'operatore a verrà letto e processato prima che venga effettuata la riduzione con b .

3.2 Automa caratteristico



3.3 Tabella di parsing

Stato	a	b	n	\$	E
0			S2		1
1	S3	S4		acc	
2	R1	R1		R1	
3			S2		5
4			S2		6
5	S3 / R2	S4 / R2		R2	
6	S3 / R3	S4 / R3		R3	

4 Esercizio 4

Soluzione

Per prima cosa generiamo una nuova grammatica che tenga conto della precedenza dell'operatore a rispetto a b . In particolare creiamo dei nuovi *non terminali*:

- E rappresenterà un'espressione intera
- T rappresenterà le operazioni con priorità maggiore
- F rappresenterà i fattori atomici o le espressioni tra parentesi

Inoltre per garantire l'associatività a sinistra costruiremo la grammatica con ricorsione a sinistra.

$$E \rightarrow EbT \mid T$$

$$T \rightarrow TaF \mid F$$

$$F \rightarrow (E) \mid id$$

Questa grammatica risolve l'ambiguità dando la precedenza all'operatore a su b ed è associativa a sinistra, ma non è LL(1), poiché presenta appunto ricorsione a sinistra.

Per eliminare la ricorsione a sinistra possiamo fattorizzare, ricordando la formula:

$$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \dots \mid \beta_1 \mid \beta_2 \mid \dots$$

$$\Downarrow$$

$$A \rightarrow \beta_1 A \mid \beta_2 A \mid \dots$$

$$A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \dots \mid \epsilon$$

Quindi applicando la formula otteniamo:

$$E \rightarrow EbT \mid T$$

$$\Downarrow$$

$$E \rightarrow TE'$$

$$E' \rightarrow bTE' \mid \epsilon$$

$$T \rightarrow TaF \mid F$$

$$\Downarrow$$

$$F \rightarrow FT'$$

$$T' \rightarrow aFT' \mid \epsilon$$

Mettendo tutto insieme otteniamo:

$$E \rightarrow TE'$$

$$E' \rightarrow bTE' \mid \epsilon$$

$$T \rightarrow FT'$$

$$T' \rightarrow aFT' \mid \epsilon$$

$$F \rightarrow (E) \mid id$$

Questa grammatica è stata vista a lezione con $b = +$ e $a = *$ e sappiamo sia LL(1).