SLR(1)

Esempio:

Data la grammatica:

E → E + T | E – T | T

T → T \* int | int

Non deve essere fattorizzata a sinistra e può essere anche ricorsiva a sinistra. (A differenza di LL(1) )

**SLR(1) Automa:**

Si aggiunge la produzione:

E' → E

e quando l'input finisce si legge il $.

L'input viene accettato dal parser SLR(1) quando, nella parsing table, finisce nella cella ACCEPT.

**Follow:**

La prima cosa da fare è calcolare l'insieme dei Follow di ogni variabile, in questo caso E', E, T:

Follow(E') = {$}

Follow(E)={+,-, $}

Follow(T)={+,-,\*, $}

**Stato iniziale:**

Lo stato iniziale del DFA è formato dalla chiusura della produzione del simbolo iniziale:

CLOSURE( E' → E ):

Dato che la prima variabile è la E, si espande E con le sue produzioni, quindi si aggiungono allo stato iniziale:

E → E + T

E → E – T

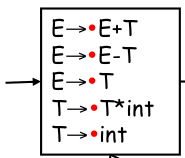
E → T

Poi si espande anche la T perché viene prodotta subito dalla E in un unico passo:

T → T \* int

T → int

In ogni produzione si mette il pallino subito dopo la freccia, quindi lo stato iniziale è:



**Transizioni:**

Adesso si guardano tutti i simboli in uscita, e per ogni simbolo si crea una transizione verso un nuovo stato del DFA.

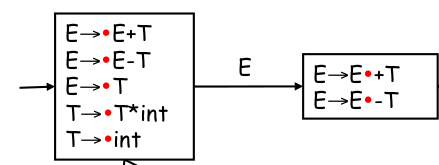
I simboli in uscita sono i primi dopo il pallino, in questo caso sono:

E, T, int.

Esce E:

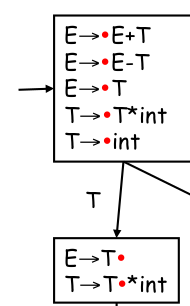
Ci sono solo le prime 2 produzioni che hanno la E dopo il pallino, quindi il nuovo stato è formato dala chiusura delle prime 2 produzioni, con il pallino spostato dopo la E.

In questo caso il + e il – non si possono espandere.



Esce T:

Si fa la chiusura della 3rd e 4th produzione, portando in avanti il pallino:



Esce int:

Si prende solo l'ultima produzione, che diventa T→int•.

------------------  
  
Se si fanno tutte queste operazioni per ogni stato, alla fine si trova l'automa SLR(1), che lista tutte le possibili configurazioni legali dello stack.

(ogni stato è accettante)

**SLR(1) Parsing Table:**

Nella tabella:

- le righe indicano gli stati del SLR(1) automa

- le colonne indicano i simboli in input (sia terminali che variabili)

Nelle colonne dei terminali sono possibili le azioni di shift e reduce, mentre nelle variabili solo le azioni di goto.

Una cella al massimo può avere una sola azione, altrimenti la grammatica non viene riconosciuta dall'SLR(1).

**Shift:**

Negli stati che hanno in uscita un terminale, si fa l'azione di shift.

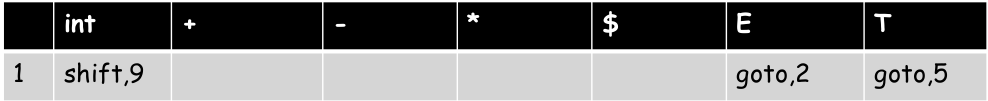
Nella cella della tabella si scrive shift, <stato destinazione>



Nel caso di prima, lo stato 1 (iniziale) aveva in uscita int verso lo stato 9, che è formato solo dalla produzione T→int•.

**Goto:**

Per ogni stato che ha in uscita una variabile, si fa goto:



Nello stato iniziale di prima uscivano E e T, vanno neggli stati 2 e 5. (righe della paring table)

**Reduce:**

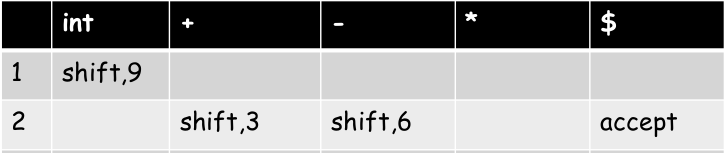
Nel caso di prima, lo stato 5 aveva una produzione con il pallino alla fine (E → T•)

Quindi, nello stato 5, si fa la Reduce da T ad E, in ogni terminale che è nei Follow(E), la variabile del Left-Hand Side:

I terminali del Follo set sono: +, -, $



**Accept:**



**Testare l'input:**

Per testare l'input, nell SLR(1) si parte sempre dallo stato iniziale e poi, ad ogni reduce, si riparte dallo stato iniziale.

L'input letto e le riduzioni vengono effettuate nello stack del parser.

Per evitare di ripartire dallo stato iniziale, si può scrivere una coppia di valori:

(nome stato), simbolo

Se si finisce in una cella bianca, durante un simbolo in input, l'input non viene accettato dalla grammatica.