

Pertanto, dopo aver scaricato I in closure(I) la domanda da porci nell'applicazione iterativa della closure è

“c’è un non-terminale dopo il punto?”

Se la risposta è NO allora ci si ferma, se è SI allora si continua “scavando” dentro quel non-terminale. Infatti il parser si aspetta come prossimi simboli tutto ciò che è derivabile da quel non-terminale.

### Funzione goto

La funzione *goto* si applica a un insieme di LR(0) items e ad un simbolo grammaticale, e produce un insieme di LR(0) items. In termini dichiarativi essa è così definita:

*goto*(I, x) è la closure dell'insieme di tutti gli LR(0) items  $A \rightarrow \alpha . x . \beta$  tali che  
 $A \rightarrow \alpha . x \beta$  è in I

Sostanzialmente, quando si applica la *goto*(I, x) si vanno a vedere in I gli LR(0) items che hanno un punto prima del simbolo x, si sposta il punto a destra di x e si applica la closure.

### *Esempio*

Per la grammatica aumentata dell'esempio precedente, se si considera

$$I = \{ E' \rightarrow E .$$

$$E \rightarrow E . + T \}$$

$$\text{allora } \text{goto}(I, +) = \{$$

$$E \rightarrow E . + T$$

$$T \rightarrow . T * F$$

$$T \rightarrow . F$$

$$F \rightarrow . ( E )$$

$$F \rightarrow . \text{id} \quad \}$$

Nota: L'applicazione di una *goto*(I, x) = I' si può rappresentare graficamente da una transizione su un grafo dal nodo I al nodo I' attraverso l'arco etichettato x.

A questo punto, il passo fondamentale della tecnica di costruzione parsing table SLR consiste nel costruire la collezione di insiemi di LR(0) items (formante gli stati del corrispondente parser) che si ottiene applicando iterativamente la funzione *goto*, e le successive closure, all'insieme di partenza:

$$I_0 = \{ \text{closure}(S' \rightarrow . S) \}$$

La funzione goto va applicata per ogni simbolo grammaticale possibile, ossia per ogni simbolo che nel corrente insieme di items ha un punto subito prima di esso. In tal modo, a partire dall'insieme iniziale  $I_0$  in cui si fa la closure dell'item iniziale  $S' \rightarrow . S$  si costruiscono proprio gli stati  $I_0 I_1 \dots I_n$  della parsing table SLR.

### Esercizio 1 (estratto da traccia d'esame):

Costruire la collezione di insiemi di LR(0) items per la grammatica context free con produzioni:

$$S \rightarrow a B c$$

$$B \rightarrow b d$$

$$B \rightarrow b$$

#### Soluzione:

Come prima cosa si considera la grammatica aumentata:

$$S' \rightarrow S$$

$$S \rightarrow a B c$$

$$B \rightarrow b d$$

$$B \rightarrow b$$

A partire da questa grammatica aumentata, si applica iterativamente la funzione goto (e le successive closure) innescandola sull'insieme di partenza  $I_0 = \{ \text{closure}(S' \rightarrow . S) \}$ . In tal modo si ottiene:

$$I_0 = \{ \text{closure}(S' \rightarrow . S) \} = \{ S' \rightarrow . S \\ S \rightarrow . a B c \}$$

$$I_1 = \text{goto}(I_0, S) = \{ S' \rightarrow S . \}$$

$$I_2 = \text{goto}(I_0, a) = \{ S \rightarrow a . B c \\ B \rightarrow . b d \\ B \rightarrow . b \}$$

$$I_3 = \text{goto}(I_2, B) = \{ S \rightarrow a B . c \}$$

$$I_4 = \text{goto}(I_3, c) = \{ S \rightarrow a B c . \}$$

$$I_5 = \text{goto}(I_2, b) = \{ B \rightarrow b . d \\ B \rightarrow b . \}$$

$$I_6 = \text{goto}(I_5, d) = \{ B \rightarrow b d . \}$$

Nota: se ogni insieme di LR(0) items  $I_i$  si disegna come un nodo di un grafo e ogni applicazione della goto  $I_k = \text{goto}(I_i, x)$  si disegna come una transizione sul grafo dal nodo  $I_i$  al nodo  $I_k$  con arco labellato  $x$ , il grafo ottenuto si chiama *goto grafo SLR* e rappresenta il diagramma stati transizione dell'automa che sta dietro il parser.

## Esercizio 2 (estratto da traccia d'esame):

Costruire la collezione di insiemi di LR(0) items per la grammatica context free con produzioni:

$$S \rightarrow A b$$

$$A \rightarrow A a$$

$$A \rightarrow a$$

### Soluzione:

Anzitutto si costruisce la grammatica aumentata:

$$S' \rightarrow S$$

$$S \rightarrow A b$$

$$A \rightarrow A a$$

$$A \rightarrow a$$

A partire da questa grammatica aumentata, si applica iterativamente la funzione goto (e le successive closure) innescandola sull'insieme di partenza  $I_0 = \{ \text{closure}(S' \rightarrow . S) \}$ . In tal modo si ottiene:

$$I_0 = \{ \text{closure}(S' \rightarrow . S) \} = \{ S' \rightarrow . S$$

$$S \rightarrow . A b$$

$$A \rightarrow . A a$$

$$A \rightarrow . a \}$$

$$I_1 = \text{goto}(I_0, S) = \{ S' \rightarrow S . \}$$

$$I_2 = \text{goto}(I_0, A) = \{ S \rightarrow A . b$$
  
$$A \rightarrow A . a \}$$

$$I_3 = \text{goto}(I_0, a) = \{ A \rightarrow a . \}$$

$$I_4 = \text{goto}(I_2, b) = \{ S \rightarrow A b . \}$$

$$I_5 = \text{goto}(I_2, a) = \{ A \rightarrow A a . \}$$

### 1.3 Algoritmo SLR

QCFG

Data una grammatica context free il passo centrale per la costruzione della sua parsing table SLR sta nella costruzione della sua collezione di insiemi di LR(0) items (o equivalentemente nella costruzione del goto grafo SLR se ne si considera la rappresentazione grafica). Questi items contengono infatti tutta l'informazione necessaria al parser per fare le sue mosse e procedere nel riconoscimento della stringa input.

Tuttavia per codificare questa informazione nel particolare formato richiesto dalle entrate della parsing table vanno applicati dei semplici passaggi di trasformazione. Diamo di seguito queste regole insieme alla linea generale dell'algoritmo

G context free  $\rightarrow$  Parsing Table SLR

### Algoritmo

L'algoritmo costruisce la grammatica aumentata di G e crea la sua parsing table SLR applicando i seguenti passi:

- 1) Anzitutto si costruisce la collezione di insiemi di LR(0) items. Gli insiemi di items  $I_0 I_1 \dots I_n$  (cioè i nodi del goto grafo) corrispondono agli stati 0, 1, ... n della parsing table. Lo stato iniziale 0 corrisponde all'insieme  $I_0 = \{ \text{closure}(S' \rightarrow . S) \}$
- 2) Per ogni transizione etichettata con un simbolo terminale  $a$ :  
 $I_k = \text{goto}(I_i, a)$   
si inserisce l'azione  $\text{shift}_k$  nell'entrata  $\text{ACTION}[i, a]$  della parsing table
- 3) Per ogni transizione etichettata con un simbolo non-terminale A:  
 $I_k = \text{goto}(I_i, A)$   
si inserisce l'azione  $\text{shift}_k$  nell'entrata  $\text{GOTO}[i, A]$  della parsing table
- 4) Per ogni insieme  $I_i$  che ha al suo interno un LR(0) item col punto alla fine  $A \rightarrow \alpha$ . si inserisce l'azione  $\text{reduce}_{A \rightarrow \alpha}$  nell'entrata  $\text{ACTION}[i, a]$  della parsing table sotto tutti i terminali  $a$  appartenenti a  $\text{Follow}(A)$
- 5) Si considera l'unico insieme  $I_i$  che contiene l'item  $S' \rightarrow S .$  e si inserisce "ACCETTA" nell'entrata  $\text{ACTION}[i, \$]$  della parsing table
- 6) Tutte le entrate della parsing table rimaste vuote si settano a "ERRORE"

Nota: Se dopo il passo 4) la parsing table in riempimento contiene almeno un conflitto, ossia un'entrata multipla del tipo shift/reduce oppure reduce/reduce nella parte action, allora il parsing fallisce e la grammatica input G è detta essere non SLR.