# Tabelle LR

Costruzione delle tabelle di parsing LR canoniche

## Precisiamo il problema di SLR

- In uno stato i la tabella indica una riduzione con una produzione $A \to \alpha$ se l'insieme di item Ii contiene l'item $A \to \alpha\bullet$ e il simbolo corrente è un $a \in$ FOLLOW(A)
- Tuttavia, in alcune situazioni, quando lo stato i appare in testa allo stack, il viable prefix $\beta\alpha$ sullo stack non può essere seguito da a in nessuna forma sentenziale destra
- Quindi la riduzione $A \to \alpha$ sarebbe sbagliata con l'input a

## Esempio

- Riconsideriamo la grammatica degli assegnamenti e delle espressioni di identificatori e puntatori
- Nello stato 2 c'era un item $R \to L\bullet$ (che corrisponde alla $A \to \alpha\bullet$ di cui sopra) e il segno = apparteneva a FOLLOW(R) (= corrisponde alla a di cui sopra)
- Ma non c'è nessuna forma sentenziale destra della grammatica che inizia per R= …
- Quindi, nello stato 2, non dovrebbe essere indicata una riduzione sul simbolo =

## Soluzione

- Aggiungiamo informazione agli stati
- Ogni stato deve contenere, oltre agli item validi, anche i simboli di input che possono seguire una handle $\alpha$ per la quale è possibile una riduzione ad A
- Gli item sono ridefiniti: sono coppie $[A \to \alpha\bullet\beta, a]$ dove $A \to \alpha\beta$ è una produzione e a è un simbolo terminale oppure \$

## Item LR(1)

- Questi item che contengono anche un simbolo terminale sono chiamati item LR(1) (1 indica che si considera un simbolo di lookahead, cioè la seconda componente è un solo simbolo)
- Il lookahead non ha effetti in un item $[A \to \alpha\bullet\beta, a]$ dove $\beta \neq \varepsilon$, ma se l'item è della forma $[A \to \alpha\bullet, a]$ allora si esegue una riduzione solo se il simbolo di lookahead è a
- In altre parole non riduciamo più per tutti i simboli di FOLLOW(A), ma per un sottoinsieme (anche proprio) di questi

## Item LR(1) validi

Formalmente, diciamo che un item LR(1) $[A \to \alpha\bullet\beta, a]$ è valido per un viable prefix $\gamma$ se e solo se c'è una derivazione

$$S \Rightarrow^*_{rm} \delta Aw \Rightarrow_{rm} \delta\alpha\beta w$$

dove

1. $\gamma = \delta\alpha$
2. o a è il primo simbolo di w oppure $w=\varepsilon$ e a=\$

## Esempio

Consideriamo la seguente grammatica:

$S \rightarrow BB$

$B \rightarrow aB \mid b$

La seguente è una derivazione rightmost:

$S \Rightarrow^*_{rm} aaBab \Rightarrow_{rm} aaaBab$

L'item $[B \rightarrow a \bullet B, a]$ è valido per un viable prefix $\gamma = aaa$ sostituendo $\delta=aa$, $A = B$, $w = ab$, $\alpha = a$ e $\beta = B$ nella definizione

## Costruzione degli item LR(1)

- La costruzione degli insiemi di item LR(1) validi è essenzialmente la stessa che per la costruzione canonica LR(0)
- Quello che cambiano sono le funzioni *closure* e *goto*.

## *Closure*

- Consideriamo un item $[A \rightarrow \alpha \bullet B\beta, a]$ nell'insieme degli item **validi** per un viable prefix $\gamma=\delta\alpha$.
- Per la validità, esiste una derivazione rightmost $S \Rightarrow^* \delta Aax \Rightarrow \delta\alpha B\beta ax$
- Supponiamo che $\beta ax$ derivi una stringa di terminali by
- Allora, per ogni produzione $B \rightarrow \eta$, abbiamo una derivazione rightmost

  $S \Rightarrow^* \delta\alpha Bby \Rightarrow \delta\alpha\eta by$
- Cioè, l'item $[B \rightarrow \bullet\eta, b]$ è valido per il viable prefix $\gamma$

## *Closure*

- Il b in questione potrebbe essere il primo carattere derivato da $\beta$, oppure, se $\beta \Rightarrow^* \varepsilon$, b potrebbe essere la a
- Per trattare uniformemente tutti i casi basta dire che b può essere un qualsiasi terminale in FIRST($\beta ax$)= FIRST($\beta a$)

## Calcolo di *closure*(I) a partire da I

**begin**
  *closure*(I) := I;
  **repeat**
    **for** each item $[A \rightarrow \alpha \bullet B\beta, a] \in$ I,
      each produzione $B \rightarrow \gamma$ in G',
      and each terminale $b \in$ FIRST($\beta a$)
      tale che $[B \rightarrow \bullet\gamma, b] \notin$ I **do**
        aggiungi $[B \rightarrow \bullet\gamma, b]$ a *closure*(I);
  **until** nessun nuovo item può essere aggiunto;
**end**;

## Esempio

$S' \rightarrow S$

$S \rightarrow CC$

$C \rightarrow cC \mid d$

Per calcolare *closure*({[$S' \rightarrow \bullet S, \$$]}) facciamo il match dell'unico item con il template $[A \rightarrow \alpha \bullet B\beta, a]$. Otteniamo $A = S'$, $\alpha=\varepsilon$, $B= S$, $\beta= \varepsilon$, $a = \$$.

Si ha FIRST($\beta a$)=FIRST($\$$)={$\$$}

Quindi aggiungiamo solo l'item $[S \rightarrow \bullet CC, \$]$

## Esempio

Continuiamo a chiudere aggiungendo tutti gli item $[C \rightarrow \bullet\gamma, b]$ con $b \in$ FIRST(C$)

Si ha che FIRST(C$)={c,d}

Quindi aggiungiamo i seguenti item:

$[C \rightarrow \bullet cC, c]$ $[C \rightarrow \bullet cC, d]$

$[C \rightarrow \bullet d, c]$ $[C \rightarrow \bullet d, d]$

Il calcolo termina dato che non c'è nessun nuovo item che fa match con il template

## Esempio

$I_0$: $[S' \rightarrow \bullet S, \$]$
$[S \rightarrow \bullet CC, \$]$
$[C \rightarrow \bullet cC, c]$
$[C \rightarrow \bullet cC, d]$
$[C \rightarrow \bullet d, c]$
$[C \rightarrow \bullet d, d]$

Abbreviazioni:

$I_0$: $S' \rightarrow \bullet S, \$$
$S \rightarrow \bullet CC, \$$
$C \rightarrow \bullet cC, c/d$
$C \rightarrow \bullet d, c/d$

## Funzione *goto*

**function** *goto*(I,X);
**begin**
sia J l'insieme di item della forma
$[A \rightarrow \alpha X \bullet \beta, a]$ tali che
$[A \rightarrow \alpha \bullet X \beta, a] \in I$
**return** *closure*(J);
**end**;

## Calcolo degli item LR(1)

- Utilizziamo la stessa identica procedura che abbiamo visto per la collezione canonica LR(0), ma con le funzioni closure e goto nuove
- Anche qui il risultato è un automa deterministico i cui stati sono insiemi di item LR(1)
- Continuiamo con l'esempio

## Esempio

- Calcoliamo tutti i *goto* per lo stato 0
- Possiamo vedere il tutto come la costruzione di un automa: *goto*($I_0$,X) è lo stato a cui si arriva partendo da 0 e facendo una transizione etichettata X
- *goto*($I_0$, S) = *closure*({$[S' \rightarrow S\bullet, \$]$})= {$[S' \rightarrow S\bullet, \$]$} = $I_1$

## Esempio

- *goto*($I_0$,C) = *closure*({$[S \rightarrow C\bullet C, \$]$}) = {$[S \rightarrow C\bullet C, \$]$,
  $[C \rightarrow \bullet cC, \$]$,
  $[C \rightarrow \bullet d, \$]$} = $I_2$
- *goto*($I_0$,c) = *closure*({$[C \rightarrow c\bullet C, c]$, $[C \rightarrow c\bullet C, d]$})= $C \rightarrow c\bullet C, c/d$
  $C \rightarrow \bullet cC, c/d$
  $C \rightarrow \bullet d, c/d$
  = $I_3$

## Esempio

- *goto*($I_0$,d) = *closure*({[C → d•, c/d]})
  = C → d•, c/d = $I_4$
- Abbiamo finito di calcolare le "transizioni uscenti" da $I_0$, passiamo ad $I_1$ = {[S' → S•, $]}
- Per $I_1$ non c'è nessuno *goto*
- Passiamo a $I_2$

## Esempio

- *goto*($I_2$,C) = *closure*({[S→C•C, $]}) = {[S→C•C, $]} = $I_5$
- *goto*($I_2$,c) = *closure*({[C→c•C, $]}) =
  C→c•C, $
  C →•cC, $
  C →•d, $ = $I_6$
- *goto*($I_2$,d) = {[C →d•, $]} = $I_7$
- Si noti che $I_7$ differisce da $I_4$ solo nelle seconde componenti

## Esempio

- Questo fenomeno è tipico per gli insiemi di item LR(1)
- Data una certa grammatica, ogni insieme della collezione canonica LR(0) corrisponde in generale all'insieme di insiemi di item LR(1) che hanno la stessa prima componente, ma seconde componenti diverse

## Esempio

- *goto*($I_3$,C) = C→cC•, c/d = $I_8$
- *goto*($I_3$,c) = *closure*({[C →c•C, c/d}) = $I_3$
- *goto*($I_3$,d) = C →d•, c/d = $I_4$
- *goto*($I_6$,C) = {C →cC•, $} = $I_9$
- *goto*($I_6$,c) = $I_6$
- *goto*($I_6$,d) = $I_7$

## Costruzione della tabella LR

- Il procedimento è lo stesso che per la costruzione della tabellaSLR, anzi, è più semplice: per le riduzioni dobbiamo guardare il simbolo di lookahead dell'item e non tutti i simboli di FOLLOW(A) se A è la parte sinistra della produzione con cui ridurre

## Algoritmo

1. Costruisci C = {$I_0$,$I_1$,...,$I_n$ }, la collezione di item LR(1)
2. Lo stato *i* del parser LR è costruito a partire da $I_i$. Le azioni di parsing per lo stato *i* sono determinate come segue:
   a) Se [A → α•aβ, b] ∈ $I_i$ e *goto*($I_i$,a) = $I_j$, allora poni *action*[*i*,a]:= "shift *j*" (a è terminale!)
   b) Se [S'→S•, $] ∈ $I_i$, allora poni *action*[*i*,$] := "accept"
   c) Se [A → α•, a] ∈ $I_i$, allora poni *action*[*i*,a] := "reduce A → α"

## Algoritmo

- Come nel caso SLR, la parte goto della tabella LR è data dalla funzione goto
- Tutte le entrate vuote corrispondono a "error"
- Lo stato iniziale del parser LR è quello costruito dall'insieme che contiene l'item $[S' \rightarrow \bullet S, \$]$ ($I_0$)
- La tabella così formata si chiama *tabella di parsing LR(1)* ***canonica***

## Tabella e conflitti

- Se nella tabella non ci sono entrate multidefinite allora la grammatica è analizzabile LR(1)
- (1) è sottinteso quando diciamo solo LR
- Un parser LR che utilizza questa tabella si chiama parser LR(1) canonico

## Esempio

- Numeriamo le produzioni della grammatica seguendo l'ordine naturale
- 0) $S' \rightarrow S$
- 1) $S \rightarrow CC$
- 2) $C \rightarrow cC$
- 3) $C \rightarrow d$
- Riempiamo la tabella

## Tabella LR(1) canonica

| STATO | | action | | goto | |
|---|---|---|---|---|---|
| | c | d | $ | S | C |
| 0 | s3 | s4 | | 1 | 2 |
| 1 | | | acc | | |
| 2 | s6 | s7 | | | 5 |
| 3 | s3 | s4 | | | 8 |
| 4 | r3 | r3 | | | |
| 5 | | | r1 | | |
| 6 | s6 | s7 | | | 9 |
| 7 | | | r3 | | |
| 8 | r2 | r2 | | | |
| 9 | | | r2 | | |

## Considerazioni

- Ogni grammatica SLR(1) è anche LR(1), ma per una grammatica SLR(1) il parser LR canonico può avere molti più stati rispetto a quelli del parser SLR
- Ad esempio, la grammatica che abbiamo usato come esempio è SLR e un parser SLR per la stessa ha 7 stati invece dei 10 del parser LR canonico appena costruito