

Analisi sintattica LL(1)

Parte seconda

Parsificazione top-down: parser LL(1)

Sia $G = \langle V, \Sigma, P, S \rangle$ una grammatica LL(1), cioè tale che per ogni coppia di produzioni a partire da uno stesso simbolo non terminale A :

$A \rightarrow \alpha$ e

$A \rightarrow \beta$, si ha

$$\text{Gui}(A \rightarrow \alpha) \cap \text{Gui}(A \rightarrow \beta) = \Phi.$$

Costruiamo a partire da G un riconoscitore che si *ispira* direttamente ad un automa a pila che accetta per stack vuoto.

Il riconoscitore scorre l'input (che sarà terminato da un simbolo di fine stringa (\$)) con una testina di lettura e usa una pila, che supporremo svilupparsi da destra a sinistra per rendere più evidente il significato delle sue mosse.

L'informazione relativa alla scelta della produzione si può condensare in una *tabella* che per ogni variabile sulla pila e ogni simbolo terminale indica la produzione (se esiste) da applicare.

Un parser LL(1) (versione iterativa) analizza una stringa leggendola una sola volta da sinistra a destra. Utilizza una tabella derivata dalla grammatica (tabella LL(1)) e una Pila.

Input: 1) Stringa da parsificare

2) Tabella **M** che memorizza, per ogni coppia
 < variabile A, simbolo in input a >
 la produzione $A \rightarrow \alpha$ se $a \in \text{Gui}(A \rightarrow \alpha)$,
 ' ' se $a \notin \text{Gui}(A \rightarrow \alpha)$.

Output: Stringa accettata o segnalazione di errore

La lettura dell'input si rappresenta di solito mediante un puntatore che punta al primo simbolo non ancora analizzato (lookahead).

Grammatiche LL(1): esempio

Una grammatica per $\{0^n 1^n \mid n > 0\}$

Si $\text{First}(S) = \{0\}$, $\text{First}(A) = \{0,1\}$, $\text{First}(S1)=\text{First}(S)=\{0\}$

Produzione	Insieme guida
$S \rightarrow 0A$	$\{0\}$
$A \rightarrow S1$	$\{0\}$
$A \rightarrow 1$	$\{1\}$

	0	1	\$
S	$S \rightarrow 0A$		
A	$A \rightarrow S1$	$A \rightarrow 1$	

La grammatica è LL(1)

Grammatiche LL(1): esempio

Input	stack	derivazione
0011\$ ↑	S	S
0011\$ ↑	0A	0A
0011\$ ↑	A	
0011\$ ↑	S1	0S1
0011\$ ↑	0A1	00A1
0011\$ ↑	A1	
0011\$ ↑	11	0011
0011\$ ↑	1	
0011\$ ↑		

Derivazione della stringa: $S \rightarrow 0A \rightarrow 0S1 \rightarrow 00A1 \rightarrow 0011$

ricorsioni sinistre

Una grammatica ricorsiva *sinistra*, cioè tale che per qualche non terminale A si ha una derivazione $A \rightarrow^+ A\alpha$, **non può essere LL(1)**.
Il caso più tipico è quando la grammatica ha una produzione $A \rightarrow A\alpha$ (ricorsione sinistra *diretta*)

Per esempio $S \rightarrow S a \mid b$

$\{b\} = \text{Gui}(S \rightarrow Sa) = \text{Gui}(S \rightarrow b)$

Derivazione tipica: per derivare la stringa ba

$S \rightarrow S a \rightarrow ba$

Nel primo passo si applica la produzione $S \rightarrow S a$
nel secondo quella $S \rightarrow b$.

Una grammatica LL(1) è:

1. Non ambigua (esiste *una sola* derivazione left-most)
2. Senza ricorsioni sinistre

ATTENZIONE: le proprietà 1, 2 sono *necessarie* ma **non** sufficienti.

Una grammatica può rispettarle tutte due ma non essere LL(1).

Esempio: $S \rightarrow aSb \mid a$

Parsificazione top-down: trasformazioni di grammatiche

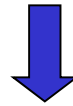
Data una grammatica non LL(1), è qualche volta possibile ottenerne una equivalente LL(1).

In particolare si puo':

1. **eliminare le ricorsioni sinistre**, sia immediate sia non immediate
2. cercare di rendere la scelta della produzione da usare ad ogni passo deterministica posticipando, quando possibile, la scelta tra diverse alternative di riscrittura che hanno un prefisso comune: **fattorizzazione sinistra**

Eliminazione delle ricorsioni sinistre immediate

$$\begin{array}{ll} A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \dots \mid A\alpha_k & k \geq 1, \quad \alpha_i \neq \varepsilon \\ A \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_h & h \geq 1 \end{array}$$



$$\begin{array}{l} A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \dots \mid \beta_h A' \\ A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \dots \mid \alpha_k A' \mid \varepsilon \end{array}$$

Infatti, per esempio:

$$A \rightarrow A\alpha_i \rightarrow A\alpha_j\alpha_i \rightarrow^* A\alpha_1 \dots \alpha_j\alpha_i \rightarrow \beta_m\alpha_1 \dots \alpha_j\alpha_i$$

si può ottenere anche da

$$A \rightarrow \beta_m A' \rightarrow \beta_m\alpha_1 A' \dots \rightarrow^* \beta_m\alpha_1 \dots \alpha_j\alpha_i A' \rightarrow \beta_m\alpha_1 \dots \alpha_j\alpha_i$$

Eliminazione delle ricorsioni sinistre

Esempio

$$E \rightarrow E + T \mid T$$

$$\begin{aligned} E &\rightarrow T E' \\ E' &\rightarrow + T E' \mid \varepsilon \end{aligned}$$

$$T \rightarrow T * F \mid F$$

$$\begin{aligned} T &\rightarrow F T' \\ T' &\rightarrow * F T' \mid \varepsilon \end{aligned}$$

$$F \rightarrow (E) \mid \text{id}$$

$$F \rightarrow (E) \mid \text{id}$$

Fattorizzazione sinistra

Per ogni non terminale A si trova il massimo prefisso α comune a due o più alternative.

Si sostituiscono tutte le produzioni:

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

con

$$\begin{aligned} A &\rightarrow \alpha A' \\ A' &\rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_m \end{aligned}$$

lasciando le altre produzioni da A inalterate.

Per esempio:

$$S \rightarrow aB \mid aC, \quad B \rightarrow b, \quad C \rightarrow c$$

diventa

$$S \rightarrow aS', \quad S' \rightarrow B \mid C, \quad B \rightarrow b, \quad C \rightarrow c$$

Calcolo di FIRST(X) (abbreviato F(X))

Un metodo per calcolare gli F(X) per una grammatica $G = \langle V, \Sigma, P, S \rangle$
dove $X \in V$:

1. Si pone inizialmente

$$F(A) = \{a \mid A \rightarrow a \alpha \in P\} \cup \{\epsilon \mid \text{se } A \rightarrow \epsilon \in P\}.$$

Altrimenti si inizializza $F(A) = \{\}$ (insieme vuoto)

2. *per ogni* produzione $A \rightarrow Y_1..Y_k$:

1. si *aggiunge* $F(Y_1) - \{\epsilon\}$ a $F(A)$.
2. se $\epsilon \in F(Y_1)$ (tipicamente se $Y_1 \rightarrow \epsilon \in P$) si *aggiunge* $F(Y_2) - \{\epsilon\}$ a $F(A)$.
3. se $\epsilon \in F(Y_1)$ e $\epsilon \in F(Y_2)$ si *aggiunge* $F(Y_3) - \{\epsilon\}$ a $F(A)$.
4. e così via finchè si trovano Y_i annullabili.
5. se $\epsilon \in F(Y_1), \dots, F(Y_k)$ si *aggiunge* ϵ a $F(A)$.

3. si ripete il passo 2, fino a che gli insiemi $F(A)$ non cambiano più

Esempio

$$G = \langle \{X, Y, Z\}, \{a, c, d\}, P, Z \rangle$$

$$\begin{aligned} P: \quad & Z \rightarrow d \mid XYZ \\ & Y \rightarrow c \mid \epsilon \\ & X \rightarrow Y \mid a \end{aligned}$$

Valori iniziali: $F(a) = \{a\}$, $F(c) = \{c\}$, $F(d) = \{d\}$ (non cambiano più)

$$F(Z) = \{d\}, \quad F(Y) = \{\epsilon, c\}, \quad F(X) = \{a\}$$

Esaminiamo le produzioni nell'ordine in cui sono scritte:

Dopo una *prima* passata: $F(Z) = \{d, c, a\}$, $F(Y) = \{\epsilon, c\}$, $F(X) = \{\epsilon, c, a\}$

Questi sono i valori *definitivi*.

Per esempio $c \in F(Z)$. Infatti $Z \rightarrow XYZ \rightarrow YYZ \rightarrow YZ \rightarrow cZ$

Esempio 2 : $F(XYZ) = \{c, a\} \cup \{c\} \cup \{d, c, a\} = \{d, c, a\}$

FIRST: Esempio 2

1. Si pone inizialmente

$$F(A) = \{a \mid A \rightarrow a \alpha \in P\} \cup \{\epsilon \mid \text{se } A \rightarrow \epsilon \in P\}.$$

Altrimenti si inizializza $F(A) = \{\}$ (insieme vuoto)

2. per ogni produzione $A \rightarrow Y_1..Y_k$:

1. si aggiunge $F(Y_1) - \{\epsilon\}$ a $F(A)$.
2. se $\epsilon \in F(Y_1)$ (tipicamente se $Y_1 \rightarrow \epsilon \in P$) si aggiunge $F(Y_2) - \{\epsilon\}$ a $F(A)$.
3. se $\epsilon \in F(Y_1)$ e $\epsilon \in F(Y_2)$ si aggiunge $F(Y_3) - \{\epsilon\}$ a $F(A)$.
4.
5. se $\epsilon \in F(Y_1), \dots, F(Y_k)$ si aggiunge ϵ a $F(A)$.

$E \rightarrow T E'$
 $E' \rightarrow + T E' \mid \epsilon$
 $T \rightarrow F T'$
 $T' \rightarrow * F T' \mid \epsilon$
 $F \rightarrow (E) \mid a$

	passo 1	passo 2	passo 3
E	{ }	{ }	{ (, a }
E'	{ +, ϵ }	{ +, ϵ }	{ +, ϵ }
T	{ }	{ (, a }	{ (, a }
T'	{ *, ϵ }	{ *, ϵ }	{ *, ϵ }
F	{ (, a }	{ (, a }	{ (, a }

FIRST di una stringa in $(\Sigma \cup V)^*$

Ricordiamo che:

$$1. F(\epsilon) = \{\epsilon\}$$

$$2. F(a\beta) = \{a\}$$

$$3. F(A\beta) = \begin{cases} F(A) & \text{se } A \text{ non è annullabile} \\ (F(A) - \{\epsilon\}) \cup F(\beta) & \text{se } A \text{ è annullabile} \end{cases}$$

Calcolo dei Follow(A) (abbreviato Fw(A))

Sia data una grammatica $G = \langle V, \Sigma, P, S \rangle$. Si eseguono i seguenti passi

1. Per ogni $A \in V$ Si calcolano $F(A)$. Si suppone inizialmente $Fw(S) = \{\$ \}$ e $Fw(A)$ vuoto per ogni altra $A \in V$.
2. per ogni produzione $A \rightarrow \alpha B \beta \in P$, e per ogni β si *aggiunge* $F(\beta) - \{\epsilon\}$ a $Fw(B)$
3. *per ogni* produzione $A \rightarrow \alpha B \beta \in P$ tale che $\epsilon \in F(\beta)$ (ovvero $\beta \in V^*$ è annullabile) si *aggiunge* $Fw(A)$ a $Fw(B)$;
4. si ripete il passo 3. fino a che gli insiemi non si cambiano più.

Note: conviene esaminare le produzioni in un ordine fisso.

Esempio

$Z \rightarrow d \mid XYZ$

$Y \rightarrow c \mid \epsilon$

$X \rightarrow Y \mid a$

Ricordiamo che $F(Z) = \{d, c, a\}$, $F(Y) = \{\epsilon, c\}$, $F(X) = \{\epsilon, c, a\}$

Valori *iniziali*: $Fw(Z) = \{\$ \}$, $Fw(Y) = \{ \}$, $Fw(X) = \{ \}$

Dopo il *passo 2.*: $Fw(Z) = \{\$ \}$, $Fw(Y) = \{d, c, a\}$, $Fw(X) = \{d, a, c\}$.

Non ci sono le condizioni per eseguire il *passo 3.* quindi si salta. Questi sono i valori definitivi.

Per esempio $a \in Fw(Y)$.

Infatti $Z \rightarrow X\textcolor{brown}{Y}Z \rightarrow X\textcolor{brown}{Y}XYZ \rightarrow X\textcolor{brown}{Y}aYZ$

FOLLOW: Esempio 2

1. Per ogni $A \in V$ Si calcolano $F(A)$. Si suppone inizialmente $Fw(S) = \{\$ \}$ e $Fw(A)$ vuoto per ogni altra $A \in V$.
2. per ogni produzione $A \rightarrow \alpha B \beta \in P$, e per ogni β si *aggiunge* $F(\beta) - \{\epsilon\}$ a $Fw(B)$
3. per ogni produzione $A \rightarrow a B \beta \in P$ tale che $\epsilon \in F(\beta)$ (ovvero $\beta \in V^*$ è annullabile) si *aggiunge* $Fw(A)$ a $Fw(B)$;
4. si ripete il passo 3. fino a che gli insiemi non si cambiano più.

$$E \rightarrow T E'$$

$$E' \rightarrow + T E' \mid \epsilon$$

$$T \rightarrow F T'$$

$$T' \rightarrow * F T' \mid \epsilon$$

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

$$F(E) = F(T) = F(F) = \{ (, a \}$$

$$F(E') = \{ +, \epsilon \}$$

$$F(T') = \{ *, \epsilon \}$$

	passo 1-2	passo 3 (prima iterazione)
E	{ \$,) }	{ \$,) }
E'	{ }	{ \$,) }
T	{ + }	{ +, \$,) }
T'	{ }	{ +, \$,) }
F	{ * }	{ *, +, \$,) }

iterando il passo 3 gli insiemi non cambiano

Parsificazione top down: esempio di Grammatica LL(1)

Espressioni aritmetiche

Produzione

Insieme guida

$$1. E \rightarrow T E'$$

$$F(TE') = F(T) = F(F) = \{ (, id \}$$

$$2. E' \rightarrow + T E'$$

$$\{ + \}$$

$$3. E' \rightarrow \epsilon$$

$$\Phi \cup FW(E') = \{), \$ \}$$

$$\left. \begin{array}{l} \{ + \} \\ \{), \$ \} \end{array} \right\} \{ + \} \cap \{), \$ \} = \Phi$$

$$4. T \rightarrow F T'$$

$$F(F) = \{ (, id \}$$

$$5. T' \rightarrow * F T'$$

$$\{ * \}$$

$$6. T' \rightarrow \epsilon$$

$$\Phi \cup FW(T') = \{ +,), \$ \}$$

$$\left. \begin{array}{l} \{ * \} \\ \{ +,), \$ \} \end{array} \right\} \{ * \} \cap \{ +,), \$ \} = \Phi$$

$$7. F \rightarrow (E)$$

$$\{ (\}$$

$$8. F \rightarrow id$$

$$\{ id \}$$

$$\left. \begin{array}{l} \{ (\} \\ \{ id \} \end{array} \right\} \{ (\} \cap \{ id \} = \Phi$$

$$FW(E') = FW(E) = \{), \$ \}$$

$$FW(T') = FW(T) = (F(E') - \{\epsilon\}) \cup FW(E) = \{ +,), \$ \}$$

Nota: 'id' è considerato un unico terminale, equivale alla 'a' delle slides precedenti

Per esempio

- $\text{FW}(E') = \{), \$\}$ perchè

$$E \rightarrow TE' \quad e$$

$$E \rightarrow^* (E) \rightarrow (TE')$$

- $\text{FW}(T') = \{+,), \$\}$ perché:

$$E \rightarrow TE' \rightarrow T$$

$$E \rightarrow^* (E) \rightarrow (TE') \rightarrow (T)$$

$$E \rightarrow^* TE' \rightarrow T + TE'$$

Parsificazione top-down: la tabella del parser LL(1)


Si possono visualizzare le produzioni da usare in funzione delle variabili e dei simboli in input con una **tabella** che contiene, per ogni coppia $\langle A, a \rangle$ la produzione $A \rightarrow \alpha$ se $a \in \text{Gui}(A \rightarrow \alpha)$, ' ' se $a \notin \text{Gui}(A \rightarrow \alpha)$.

Nel caso delle espressioni aritmetiche:

	()	+	*	id	\$
E	$E \rightarrow T E'$				$E \rightarrow T E'$	
E'		$E' \rightarrow \epsilon$	$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$				$T \rightarrow F T'$	
T'		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$
F	$F \rightarrow (E)$				$F \rightarrow \text{id}$	

La grammatica è LL(1) in quanto in ogni elemento della tabella compare al massimo una produzione: gli insiemi guida delle riscritture di una stessa variabile sono disgiunti.

Parsificazione top-down: esempio di analisi per espressioni aritmetiche

input	stack 	derivazione
(id+id)*id- ↑	E	E
(id+id)*id- ↑	TE'	TE'
(id+id)*id- ↑	FT'E'	FT'E'
(id+id)*id- ↑	(E)T'E'	(E)T'E'
(id+id)*id- ↑	E)T'E'	(E)T'E'
(id+id)*id- ↑	TE')T'E'	(TE')T'E'
(id+id)*id- ↑	FT'E')T'E'	(FT'E')T'E'
(id+id)*id- ↑	idT'E')T'E	(id T'E')T'E'
(id+id)*id- ↑	T'E')T'E	(id T'E')T'E'
(id+id)*id- ↑	E')T'E	(id E')T'E'
(id+id)*id- ↑	+TE')T'E	(id +T E')T'E'

NOTA: leggasì \$ al posto di --|

Parsificazione top-down: esempio di analisi

input	stack	derivazione
(id+id)*id- ↑	TE')T'E	(id +T E')T'E'
(id+id)*id- ↑	FT')T'E'	TE'
.....
(id+id)*id- ↑	T'E'	(id + id)T'E'
(id+id)*id- ↑	*FT'E'	(id + id)*FT'E'
(id+id)*id- ↑	FT'E'	(id + id)* FT'E'
(id+id)*id- ↑	id T'E'	(id + id)* id T'E'
(id+id)*id- ↑	T'E'	(id + id)* id T'E'
(id+id)*id- ↑	E'	(id + id)* id E'
(id+id)*id- ↑		(id + id)* id

NOTA: leggasi \$ al posto di --|

1. Eliminare la ricorsione sinistra dalle grammatiche:

$$S \rightarrow R \mid 0$$

$$R \rightarrow RC \mid R0 \mid C$$

$$C \rightarrow 1 \mid 2 \mid 3$$

$$S \rightarrow S s- \mid S\{S\} \mid D \mid \varepsilon$$

$$D \rightarrow TL-R$$

$$R \rightarrow \varepsilon \mid D$$

$$L \rightarrow L ; m \mid m$$

$$T \rightarrow i \mid r$$

N.B. $\{s, -, \{, \}, ;, m, i, r\}$
è l'insieme dei simboli
terminali

2. Data la seguente grammatica:

$$A \rightarrow A s B$$

$$A \rightarrow A m B$$

$$A \rightarrow B$$

$$A \rightarrow a$$

$$B \rightarrow (A)$$

$$B \rightarrow b$$

- Eliminare le ricorsioni sinistre
- Costruire sia nella grammatica data, sia nella grammatica ottenuta senza ricorsioni sinistre, l'albero di derivazione per la stringa:
 $b m ((a s b) s b)$

Parsificazione top-down: esercizi

1. Data la grammatica con il seguente insieme di produzioni:
 $\{S \rightarrow RA, S \rightarrow A[S], R \rightarrow E = B, B \rightarrow b, E \rightarrow bA, A \rightarrow \epsilon\}$
 - a) Calcolare gli inizi (first) e i seguiti (follow) dei simboli non terminali;
 - b) Dire se la grammatica è LL(1), motivando la risposta.
2. Per ognuna delle seguenti grammatiche, specificate dall'insieme delle produzioni, costruire gli insiemi guida delle produzioni e, se la grammatica è LL(1), scrivere la tabella di parsificazione.

$G_1:$ $N \rightarrow D K$
 $K \rightarrow N$
 $K \rightarrow \epsilon$
 $D \rightarrow 0$
 $D \rightarrow 1$

$G_2:$ $E \rightarrow E + T \mid T$
 $T \rightarrow T * F \mid F$
 $F \rightarrow (E) \mid id$