UNIVERSITÀ
DEGLI STUDI
DI TORINO

Grammatiche LL(1):
Insiemi guida e ricorsione sinistra

a.a. 2018-2019

Riferimenti bibliografici

Le slide seguenti presentano, in modo più dettagliato, gli argomenti trattati sul testo:

Compilatori: principi, tecniche e strumenti, A.V. Aho, M. S. Lam, R. Sethi, J.D. Ullman

Analisi sintattica [cap. 4]

4.3.3 Eliminazione della ricorsione sinistra
(solo ricorsione immediata)
4.4.2 Funzioni FIRST e FOLLOW

Parsing look ahead 1: insieme FIRST

Per definire gli insiemi guida delle produzioni di una grammatica servono le nozioni di **FIRST** (inizi) e **FOLLOW** (seguiti).

Data una grammatica $G = \langle V, \Sigma, P, S \rangle$, l'insieme **FIRST** (insieme degli inizi) di una stringa α di variabili e terminali è l'insieme dei terminali con cui iniziano le stringhe derivabili da α nella grammatica G.

$$\boxed{\text{FIRST}(\alpha) = \{a \mid \alpha \Rightarrow^* a\beta\} \cup \{\varepsilon \mid \text{se } \alpha \Rightarrow^* \varepsilon\}}$$

Parsing look ahead 1: insieme FIRST

$$\text{FIRST}(\alpha) = \{a \mid \alpha \Rightarrow^* a\beta\} \cup \{\varepsilon \mid \text{se } \alpha \Rightarrow^* \varepsilon\}$$

$F(\alpha)$ soddisfa queste regole (ricorsive):

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

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

3. $F(A\beta) = \begin{cases} F(A) & \text{se } \varepsilon \notin F(A) \\ (F(A) - \{\varepsilon\}) \cup F(\beta) & \text{se } \varepsilon \in F(A) \end{cases}$

dove $F(A) = \bigcup_{i=1}^{K} F(\gamma_i)$

Se $A \rightarrow \gamma_1 \mid \gamma_2 \mid \ldots \mid \gamma_k$ sono le produzioni della grammatica con A come variabile di testa

Parsing look ahead 1: insieme FOLLOW

Data una grammatica $G = \langle V, \Sigma, P, S \rangle$, l'insieme **FOLLOW** (insieme dei <u>seguiti</u>) di una variabile A è l'insieme dei terminali con cui iniziano le stringhe che seguono A nelle forme sentenziali della grammatica G. Formalmente:

$$\boxed{FOLLOW(A) = \{a \mid S \Rightarrow^* \alpha A a \beta\} \cup \{\$ \mid \text{se } S \Rightarrow^* \alpha A\}}$$

FOLLOW(A) (FW(A)) soddisfa la seguente equazione:

$$
\begin{aligned}
FW(A) = \;& \{F(\beta) - \{\varepsilon\} \mid B \to \alpha A \beta \in P\} \cup \\
& \{FW(B) \mid B \neq A \text{ e } B \to \alpha A \beta \in P \;\&\; \varepsilon \in F(\beta)\} \cup \\
& \{\$ \mid A \text{ è lo start symbol di } G\}
\end{aligned}
$$

Parsing predittivo: insieme FOLLOW

Un metodo per calcolare i follow delle variabili di una grammatica

$G = \langle V, \Sigma, \mathbb{P}, S \rangle$:

1. Per ogni $A \in V$ si pone $FW(A) = \{ \}$

2. per *ogni* produzione $B \to \alpha A \beta \in \mathbb{P}$, si *aggiunge* a FW (A) l'insieme $F(\beta) - \{\varepsilon\}$;

3. per *ogni* produzione $B \to \alpha A \beta \in \mathbb{P}$ tale che $\varepsilon \in F(\beta)$ (ovvero $\beta \in V^*$ è annullabile) si *aggiunge* a FW(A) l'insieme FW(B).

Parsing look ahead 1: insieme GUIDA

Data una grammatica G, *l'insieme GUIDA* di una produzione della grammatica $A \to \alpha$ (*Gui*($A \to \alpha$)) è l'insieme dei simboli terminali con cui iniziano le stringhe generate da α (F(α)) e, nel caso in cui α si riduca alla stringa vuota ε, dei simboli terminali con cui iniziano le stringhe che seguono A nelle forme sentenziali della grammatica (FW(A)).

$$
\boxed{
\begin{array}{c}
Gui(A \to \alpha) = \{a \mid S \underset{lm}{\Rightarrow}^{+} wA\beta \underset{lm}{\Rightarrow} w\alpha\beta \underset{lm}{\Rightarrow}^{*} wa\gamma\} \cup \\
\{\$ \mid S \underset{lm}{\Rightarrow}^{+} wA\beta \Rightarrow^{+} wA\}
\end{array}
}
$$

<u>*Nota*</u>*:* $a\gamma$ ($a \in \Sigma$) può essere:

- $a\alpha'\beta$ se $A \Rightarrow^* \alpha \Rightarrow^* a\alpha'$ (inizi di α) oppure
- $a\beta'$ se $A \Rightarrow^* \varepsilon$ e $\beta \Rightarrow^* a\beta'$ (seguiti di A)

Parsing look ahead 1: insieme GUIDA

Gui$(A \to \alpha)$ soddisfa la seguente equazione:

$$Gui(A \to \alpha) = \begin{cases} F(\alpha) & \text{se } \varepsilon \notin F(\alpha) \\ (F(\alpha) - \{\varepsilon\}) \cup FW(A) & \text{se } \varepsilon \in F(\alpha) \end{cases}$$

Una *grammatica è LL(1)* se per ogni non terminale A e per ogni coppia di produzioni $A \to \alpha$ e $A \to \beta$, gli insiemi guida sono disgiunti:

$$Gui(A \to \alpha) \cap Gui(A \to \beta) = \Phi$$

Parsing look ahead 1: insieme GUIDA

<u>Esempio</u>

Produzioni

$A \to PR$	*Gui*($A \to PR$) = {a, *, c}
$P \to aP$	*Gui*($P \to aP$) = {a}
$P \to \varepsilon$	*Gui*($P \to \varepsilon$) = {*, c}
$R \to QS$	*Gui*($R \to QS$) = {*, c}
$S \to bQS$	*Gui*($S \to bQS$) = {b}
$S \to \varepsilon$	*Gui*($S \to \varepsilon$) = {%, $}
$Q \to *A\%$	*Gui*($Q \to *A\%$) = {*}
$Q \to c$	*Gui*($Q \to c$) = {c}

Parsing look ahead 1: insieme GUIDA

Esempio

Produzioni

$Z \rightarrow d$	*Gui*$(Z \rightarrow d) = F(d) = \{d\}$
$Z \rightarrow XYZ$	*Gui*$(Z \rightarrow XYZ) = F(XYZ) = (F(X) - \{\varepsilon\}) \cup F(YZ) = \{a, c, d\}$
$Y \rightarrow c$	*Gui*$(Y \rightarrow c) = \{c\}$
$Y \rightarrow \varepsilon$	*Gui*$(Y \rightarrow \varepsilon) = (F(\varepsilon) - \{\varepsilon\}) \cup FW(Y) = \{a, c, d\}$
$X \rightarrow Y$	*Gui*$(X \rightarrow Y) = (F(Y) - \{\varepsilon\}) \cup FW(X) = \{a, c, d\}$
$X \rightarrow a$	*Gui*$(X \rightarrow a) = \{a\}$

$F(Y) = F(c) \cup F(\varepsilon) = \{c, \varepsilon\}$
$F(X) = F(a) \cup F(Y) = \{a, c, \varepsilon\}$
$F(Z) = \{d\} \cup F(XYZ) = \{d\} \cup (F(X) - \{\varepsilon\}) \cup F(YZ) =$
$= \{a, c, d\} \} \cup (F(Y) - \{\varepsilon\}) \cup \{d\} = \{a, c, d\}$

$FW(Z) = \{\$\}$
$FW(Y) = F(Z) \cup FW(X) = \{a, c, d\}$
$FW(X) = F(YZ) = (F(Y) - \{\varepsilon\}) \cup F(Z) = \{a, c, d\}$

Parsing look ahead 1: insieme GUIDA

Esempio

Grammatica	Insiemi guida
1. $S \rightarrow PQ$	{a, b, c}
2. $Q \rightarrow \&PQ$	{&}
3. $Q \rightarrow \varepsilon$	{$}
4. $P \rightarrow aPb$	{a}
5. $P \rightarrow bPa$	{b}
6. $P \rightarrow c$	{c}

$\Rightarrow$ Grammatica LL(1)

Analizzatore a discesa ricorsiva: esempio

Grammatica	Insiemi guida
1 $S \rightarrow a\ B\ b$	{a}
2 $S \rightarrow B\ S$	{b, c}
3 $B \rightarrow b$	{b}
4 $B \rightarrow c$	{c}

→ Grammatica LL(1)

```
main discesa_ricorsiva( )
        cc ← PROSS
        S( )
        if (cc = '$') "stringa accettata"
        else ERRORE(…)
```

```
function S( )
        if (cc = 'a')
                cc ← PROSS
                B( )
                if (cc = 'b') cc ← PROSS
                else ERRORE(…)
        else if (cc = 'b' or cc = 'c')
                B( )
                S( )
        else ERRORE(…)
```

```
function B( )
        if (cc = 'b')  cc ← PROSS
        else if (cc = 'c')  cc ← PROSS
        else ERRORE(…)
```

Ricorsione sinistra

Una grammatica ricorsiva sinistra, cioè tale che per qualche non terminale A si ha una derivazione $A \Rightarrow^+ A\alpha$, non è LL(1).

$S \rightarrow S\ a \mid b$

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

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

Gui$(T \rightarrow T * F)$ = *Gui*$(T \rightarrow F) = \{ (, id \}$
Gui$(E \rightarrow E + T)$ = *Gui*$(E \rightarrow T) = \{ (, id \}$

Proprietà delle grammatiche LL(1)

1. Non ambigua

2. Senza ricorsioni sinistre

3. Per ogni coppia di produzioni del tipo:

   $A \rightarrow \alpha$

   $A \rightarrow \beta$

   - da α e β non derivano stringhe che iniziano con lo stesso terminale: $F(\alpha) \cap F(\beta) = \Phi$.
   - al più una tra α e β è annullabile.
   - se α è annullabile ($\alpha \Rightarrow^* \varepsilon$), da β non deriva nessuna stringa che inizia con un terminale nell'insieme FW(A):
     $F(\beta) \cap FW(A) = \Phi$.

Eliminazione della ricorsione sinistra immediata

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

a) Eliminazione delle ricorsioni sinistre immediate

$$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_k \qquad k \geq 1,\ \alpha_i \neq \varepsilon$$
$$A \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_h \qquad h \geq 1$$

$$\Downarrow$$

$$A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_h A'$$
$$A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_k A' \mid \varepsilon$$

Eliminazione della ricorsione sinistra immediata: esempi

	Grammatica con ricorsione sinistra		Grammatica equivalente con ricorsione destra	
a)	$S \rightarrow Sab \mid Sba \mid c$		$S \rightarrow cS'$ $S' \rightarrow abS' \mid baS' \mid \varepsilon$	
b)	$E \rightarrow E + T$	{(, id}	$E \rightarrow T\ E'$	{(, id}
	$E \rightarrow T$	{(, id}	$E' \rightarrow +\ T\ E'$	{+}
			$E' \rightarrow \varepsilon$	{), $}
	$T \rightarrow T * F$	{(, id}	$T \rightarrow F\ T'$	{(, id}
	$T \rightarrow F$	{(, id}	$T' \rightarrow *\ F\ T'$	{* }
			$T' \rightarrow \varepsilon$	{+,), $}
	$F \rightarrow (\ E\ )$	{(}	$F \rightarrow (\ E\ )$	{(}
	$F \rightarrow id$	{id}	$F \rightarrow id$	{id}

Eliminazione della ricorsione sinistra non immediata: un esempio

b) Eliminazione delle ricorsioni sinistre non immediate

$$\begin{array}{l} A_1 \to A_2\, a \\ A_2 \to A_3\, b \\ A_3 \to A_1\, c \mid d \end{array} \longrightarrow \begin{array}{l} A_1 \to A_2\, a \\ A_2 \to A_3\, b \\ A_3 \to A_2\, a\, c \mid d \end{array} \longrightarrow \begin{array}{l} A_1 \to A_2\, a \\ A_2 \to A_3\, b \\ A_3 \to A_3\, b\, a\, c \mid d \end{array}$$

Eliminazione ricorsione immediata

$$\longrightarrow \begin{array}{l} A_1 \to A_2\, a \\ A_2 \to A_3\, b \\ A_3 \to d\, A_3' \\ A_3' \to b\, a\, c\, A_3' \mid \varepsilon \end{array}$$

Esempio: espressioni aritmetiche

$A \rightarrow A\alpha_1 | A\alpha_2 | \ldots | A\alpha_k$
$A \rightarrow \beta_1 | \beta_2 | \ldots | \beta_h$

$\Rightarrow$

$A \rightarrow \beta_1 A' | \beta_2 A' | \ldots | \beta_h A'$
$A' \rightarrow \alpha_1 A' | \alpha_2 A' | \ldots | \alpha_k A' | \varepsilon$

$E \rightarrow E + T$
$E \rightarrow E - T$
$E \rightarrow T$

$\Rightarrow$

$E \rightarrow T E'$
$E' \rightarrow + T E'$
$E' \rightarrow - T E'$
$E' \rightarrow \varepsilon$

$T \rightarrow T * F$
$T \rightarrow T / F$
$T \rightarrow F$

$\Rightarrow$

$T \rightarrow F T'$
$T' \rightarrow * F T'$
$T' \rightarrow / F T'$
$T' \rightarrow \varepsilon$

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

$\Rightarrow$

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

Esempio: espressioni aritmetiche

Produzioni	Insiemi guida	
1. $E \rightarrow T\ E'$	$F(TE') = F(T) = F(F) = \{\ (,\ a\}$	
2. $E' \rightarrow +\ T\ E'$	$\{ + \}$	
3. $E' \rightarrow -\ T\ E'$	$\{ - \}$	$\{ + \} \cap \{ - \} \cap \{\ ),\ \$\} = \Phi$
4. $E' \rightarrow \varepsilon$	$FW(E') = \{\ ),\ \$\}$	
5. $T \rightarrow F\ T'$	$F(F) = \{(,\ a\}$	
6. $T' \rightarrow *\ F\ T'$	$\{ * \}$	
7. $T' \rightarrow /\ F\ T'$	$\{ / \}$	$\{ * \} \cap \{ / \} \cap \{ +,\ -,\ ),\ \$\} = \Phi$
8. $T' \rightarrow \varepsilon$	$FW(T') = \{ +,\ -,\ ),\ \$\}$	
9. $F \rightarrow (E)$	$\{ (\}$	$\{ (\} \cap \{ a \} = \Phi$
10. $F \rightarrow a$	$\{ a \}$	

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

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

Esempio

Tabella

	(	)	+	-	*	/	a	$
E	$E \rightarrow T\,E'$						$E \rightarrow T\,E'$	
E'		$E' \rightarrow \varepsilon$	$E' \rightarrow +T\,E'$	$E' \rightarrow -T\,E'$				$E' \rightarrow \varepsilon$
T	$T \rightarrow F\,T'$						$T \rightarrow F\,T'$	
T'		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$	$T' \rightarrow *FT'$	$T' \rightarrow /\,FT'$		$T' \rightarrow \varepsilon$
F	$F \rightarrow (E)$						$F \rightarrow a$	

La grammatica è LL(1) in quanto in ogni elemento della tabella compare al massimo una produzione.

Esempio: espressioni aritmetiche

1. $E \rightarrow T\ E'$
2. $E' \rightarrow +\ T\ E'$
3. $E' \rightarrow -\ T\ E'$
4. $E' \rightarrow \varepsilon$
5. $T \rightarrow F\ T'$
6. $T' \rightarrow *\ F\ T'$
7. $T' \rightarrow /\ F\ T'$
8. $T' \rightarrow \varepsilon$
9. $F \rightarrow (\ E\ )$
10. $F \rightarrow a$

<expr> ::= <term> <exprp>

<exprp> ::= + <term> <exprp>
| - <term> <exprp>
| ε

<term> ::= <fact> <termp>

<termp> ::= * <fact> <termp>
| / <fact> <termp>
| ε

<fact> ::= (<expr>)
| NUM

Esempio: espressioni aritmetiche

<expr> ::= <expr> + <term>
| <expr> - <term>
| <term>

<term> ::= <term> * <fact>
| <term> / <fact>
| <fact>

<fact> ::= (<expr>) | NUM

<expr> ::= <term> <exprp>

<exprp> ::= + <term> <exprp>
| - <term> <exprp>
| ε

<term> ::= <fact> <termp>

<termp> ::= * <fact> <termp>
| / <fact> <termp>
| ε

<fact> ::= (<expr>) | NUM

Esercizi

Per ognuna delle seguenti grammatiche, specificate dall'insieme delle produzioni, verificare che sia LL(1) e costruire l'analizzatore a discesa ricorsiva.

G_1:
$N \to D\ K$
$K \to N$
$K \to \varepsilon$
$D \to 0$
$D \to 1$

G_2:
$S \to F$
$F \to$ «P[F]
$F \to P$
$P \to cP$
$P \to \#$

G_3:
$L \to S\ T$
$T \to S\ T$
$T \to \varepsilon$
$S \to id := E;$
$E \to id\ G$
$G \to +\ id\ G$
$G \to \varepsilon$

G_4:
$S \to RA$
$S \to A[S]$
$R \to E = B$
$B \to b$
$E \to bA$
$A \to \varepsilon$

G_5:
$S \to 0S \mid 1S \mid 0A \mid 1B$
$A \to 0A \mid 1B \mid \varepsilon$
$B \to 1A \mid 0B \mid \varepsilon$

G_6:
$S \to a\ B\ b$
$S \to B\ S$
$B \to b$
$B \to c$

Esercizi

1. Data la grammatica con il seguente insieme di produzioni:
   $\{S \to RA,\ S \to A[S],\ R \to E = B,\ B \to b,\ E \to bA,\ A \to \varepsilon\}$
   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 l'algoritmo che implementa l'automa look ahead 1.

   G_1: $N \to D\ K$
   $K \to N$
   $K \to \varepsilon$
   $D \to 0$
   $D \to 1$

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

Esercizi

1. Eliminare la ricorsione sinistra dalle grammatiche con le produzioni seguenti:

$\mathcal{P}_1$: $S \rightarrow R \mid 0$
$R \rightarrow RC \mid R0 \mid C$
$C \rightarrow 1 \mid 2 \mid 3$

$\mathcal{P}_2$: $S \rightarrow Ss\text{-} \mid S\{S\} \mid D \mid \varepsilon$
$D \rightarrow TL\text{-}R$
$R \rightarrow \varepsilon \mid D$
$L \rightarrow L;m \mid m$
$T \rightarrow i \mid r$

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

2. Data la grammatica con le seguenti produzioni :

$A \rightarrow A\ s\ B$
$A \rightarrow A\ m\ B$
$A \rightarrow B$
$A \rightarrow a$
$B \rightarrow (A)$
$B \rightarrow b$

a) Eliminare le ricorsioni sinistre
b) Costruire sia nella grammatica data, sia nella grammatica ottenuta senza ricorsioni sinistre, l'albero di derivazione per la stringa: bm((asb)sb)