LFC (Linguaggi Formali e Compilatori) - Note del Corso

Edoardo Lenzi

November 18, 2017

Contents

Chapter 1

Analisi Sintattica

$S \to cAd$ $A \to ab|a$

1.1 Parsing Top-down

Parto dal starting symbol ed espando le derivazioni dando prioritá alle derivazioni piú a sinistra. Cerco quindi di ricostruire una derivazione leftmost della stringa w data in input.

$w\$, \$ \notin V$

$w = cabd$

Per ricostruire la parola w parto dalla prima derivazioine $S \to cAd$ derivo la A piú a sinistra (leftmost) e posso scegliere fra a ed ab; scelgo a e mi accorgo che ho sbagliato, torno in dietro e scelgo ab.

1.2 Parsing Top-down predittivo (o non ricorsivo)

Cambio la grammatica sopra in: $S \to cAd$ $A \to aB$ $B \to b|\epsilon$ Cosí non sbaglio

1.2.1 Grammatica LL(1)

prima L: leggiamo la input string da sinistra
seconda L: ricostruiamo una leftmost derivazione
(1): decidiamo quale operazione effettuare guardando un solo simbolo in input
Le grammatiche LL(1) sono un subset delle grammatiche libere.

1.3 Algoritmi di Parsing

input buffer	w$
stack	bottom[$] top
parsing table	con tante righe quante non terminali, tante colonne quante terminali ($ incluso) in ogni cella metto un'eventuale trasformazione o "error "

1.3.1 Algoritmo di parsing non-ricorsivo

input	stringa w, tabella parsing non ricorsivo T, per G
output	derivazione leftmost di w se $w \in L(G)$, error() altrimenti

---

```
//init
buffer = {w$};
stack.push($S);

let b il primo simbolo di w
let x il top dello stack

while(x != $){
    if(x == b){
        pop(x);
        let b il simbolo necessario di w;
    } else if(x e'' terminale){
```

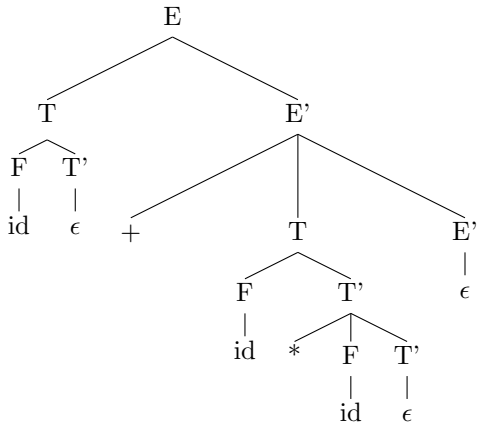
```
        error();
    } else if(T[x,b] contiene X -> Y1...Yn){
        return X -> Y1...Yn;
        pop(x);
        push(Yn...Y1)
    }
    let x il top dello stack
}
```

1.3.2 Esempio

$E \to TE'$
$E' \to +TE'|\epsilon$
$T \to FT'$
$T' \to *FT'|\epsilon$
$F \to id$

	id	+	*	$
E	$E \to TE'$			
E'		$E' \to TE'$		$E' \to \epsilon$
T	$T \to FT'$			
T'		$T' \to \epsilon$	$T' \to *FT'$	$T' \to \epsilon$
F	$F \to id$			

pila	input	output
$\underline{\$E}$	$\underline{id}$+id*id$	$E \to TE'$
$ E $\underline{T'}$		$E \to TE'$
$\$ET'\underline{F}$		$F \to id$
$ ET' $\underline{id}$		
$\$E'\underline{T''}$	+$\underline{id}$*id $	$T'' \to \epsilon$
$\underline{\$E'}$		$E' \to TE'$
$ E'T $\underline{+}$	$\underline{id}$*id$	
$ E'$\underline{T}$		
	...Avanti cosı	



1.3.3 Esercizio

$S \to aA|bB$
$A \to c$
$B \to d$

$w = ac\$$

Parsing: $S \implies aA \implies ac$

Nella tabella metto le produzioni della grammatica:

	a	b	c	d	$
S	$S \to aA$	$S \to bB$			
A			$A \to c$		
B				$B \to d$	

1.4 Calcolo First

Data una generica $\alpha \in V^*$ per G=(V, T, S, P), first(α) é l'insieme dei simboli terminali b tali che $\alpha \implies bv$. Inoltre se $\alpha \implies \epsilon$ allora $\epsilon \in first(\alpha)$

1.4.1 Esercizio

$S \to A|B$
$A \to a|C$
$C \to \epsilon$
Allora first(A) = $\{a, \epsilon\}$ (ϵ perché posso fare $A \implies C \implies \epsilon$).

1.4.2 Esercizio

$S \to A|B$
$A \to a|C$
$C \to bB$
Allora first(A) = $\{a, b\}$ (b perché posso fare $A \implies C \implies bB$, ma B non esiste).

1.4.3 Esercizio

$S \to A|B$
$A \to a|C$
$C \to bB$
$B \to c$
Allora first(A) = $\{a, b\}$ ($A \implies C \implies bB \implies bc$, ma tengo solo il primo simbolo (b))

1.4.4 Esercizio

$A \to A|C$
$C \to bB|\epsilon$
$B \to c$
Allora first(A) = $\{a, b, \epsilon\}$

G=(V,T,S,P) Sia $X \in V$. L'insieme first(X) viene calcolato come segue:

1) inizializzo first(X) vuoto $\forall\ X \in V$

2) se $X \in T$ allora first(X) = $\{X\}$

3) se $X \to \epsilon \in P$ allora aggiungere ϵ ai first(X)

4) se $X \to Y_1...Y_n \in P$, con $n \geq 1$ allora uso la seguente procedura:

```
j = 1;
while(j <= n){
    aggiungere ai first(X) ogni b tale che b in first(Yj)
    if(epsilon in first(Yj)){
        j++;
    } else {
        break;
    }
}

if(j == n+1){
    aggiungere epsilon ai first(X);
}
```