Grammatiche LL(1)

Giovedì 15 Novembre

Grammatica LL(1)

Una grammatica è LL(1) se e solo se per ogni produzione del tipo A->$\alpha|\beta$ si ha:

- α e β non derivano stringhe che cominciano con lo stesso simbolo a.
- Al più uno tra i due può derivare la stringa vuota.
- Se β->*ε allora α non deriva stringhe che cominciano con terminali che stanno in FOLLOW(A). Analogamente per α

Equivalentemente affinché una grammatica sia LL(1) deve avvenire che per ogni coppia di produzioni A->$\alpha|\beta$

1. FIRST(α) e FIRST(β) devono essere disgiunti
2. Se ε è in FIRST(β) allora FIRST(α) e FOLLOW (A) devono essere disgiunti.

Come costruire la tabella?

1. Per ogni regola X -> α di G, si inserisce nella casella (X, **t**) la regola X -> α, per ogni **t** tale che **t** $\in$ **FIRST**(α)
2. Per ogni regola X -> α di G, per cui $\alpha \Rightarrow^* \varepsilon$ ($\varepsilon \in$ **FIRST**(α)), si inserisce nella casella (X, **t**) la regola X -> α , per ogni **t** tale che **t** $\in$ **FOLLOW**(X). Se $\varepsilon \in$ **FIRST**(α) and $\$ \in$ **FOLLOW**(X), si inserisce la regola X -> α in (X,$).
3. Le caselle non definite definiscono un errore.

NOTA: Se G è ricorsiva sinistra o ambigua, la tabella avrà caselle con valori multipli.

Altra definizione di Grammatica LL(1)

*Una grammatica la cui tabella **LL(1)** non contiene più di un elemento nelle caselle è detta **LL(1)***

Osservazione: per costruzione una grammatica **LL(1)** *non è ambigua, né ricorsiva sinistra*

Esercizio: esempi di grammatiche LL(1) e non

- G: S->aSb|ε
- G: S->+SS| *SS| id
- G: S->aSb|aSc| ε
- G: S->aSa | bSb | a | b
- G: S->iEtS|iEtSeS|a
  E->b

G: S->aSb|ε

FIRST(S)={a, ε}, FIRST(a)={a}, FIRST(b)={b}

FOLLOW(S)={b,$}

	a	b	$
S	S->aSb	S->ε	S->ε

G: S->+SS| *SS| id

Per semplicità consideriamo solo i simboli non terminali

FIRST(S)={+, *, id}

FOLLOW(S)={$,+,*,id}

	+	*	id	$
S	S->+SS	S->*SS	S->id	

G: S->aSb|aSc| ε

Metodo della fattorizzazione sinistra:

G’: S->aSA| ε

A->b | c

Per semplicità consideriamo solo i simboli non terminali

FIRST(S)={a, ε}   FIRST(A)={b,c}

FOLLOW(S)={$,b,c}=FOLLOW(A)

	a	b	c	$
S	S->aSA	S->ε	S->ε	S->ε
A		A->b	A->c	

G: S->aSa | bSb | a | b

Metodo della fattorizzazione sinistra:

G': S->aA| bB

A->Sa | ε

B->Sb | ε

Per semplicità consideriamo solo i simboli non terminali

FIRST(S)={a, b}   FIRST(A)={a,b, ε}=FIRST(B)

FOLLOW(S)={a,b,$}=FOLLOW(A)=FOLLOW(B)

	a	b	$
S	S->aA	S->bB	
A	A->Sa A->ε	A->Sa A->ε	A->ε
B	B->Sb B->ε	B->Sb B->ε	B->ε

Non è un linguaggio LL(1)

G: S->iEtS|iEtSeS|a
E->b

Fattorizzazione sinistra:

G': S->iEtSS'|a

S'->eS|ε

E->b

	a	b	e	i	t	$
S	S->a			S->iEtSS'		
S'			S'->ε S'->eS			S'->ε
E		E->b				

Come ottenere grammatiche LL(1)

- Verificare, se è possibile, che non sia ambigua. In caso contrario, se si può si rimuova l'ambiguità;

- Controllare che non presenti ricorsioni sinistre. In caso contrario trasformarle in ricorsioni destre.

- Se un simbolo non terminale ammette più derivazioni con lo stesso prefisso applicare il metodo della fattorizzazione sinistra.

- In alternativa, può essere necessario allungare la lunghezza della prospezioni. (Equivale a considerare parser LL(k), k>1)

Limiti della famiglia LL(k)

- Non tutti i linguaggi verificabili da parser deterministici sono generabili da grammatiche LL(k).

Per esempio:
$L=\{a^*a^nb^n \mid n>=0\}$ Linguaggio deterministico ma non LL(k)

È generato dalla grammatica
S-> A| aS
A-> aAb |ε

Altro esempio:
S-> R | (S)
R-> E=E
E-> a | (E+E)
Definisce relazioni di uguaglianza tra espressioni aritmetiche additive.
Una stringa che inizia con "(" può essere una relazione R parentesizzata oppure un'espressione.
Il linguaggio non è LL(k) ma è deterministico.

Relazione tra grammatiche LL(k)

E' possibile generalizzare la nozione di **FIRST** nel parsing predittivo in modo da restituire i primi ***k*** token in input, e costruire una tabella predittiva in cui le righe sono i simboli non-terminali e *le colonne sono sequenze di **k** terminali* .

Le grammatiche non ambigue corrispondenti sono le ***LL(K)***