Parsing Top-Down e Grammatiche LL

PROBLEMA:

- Data $G = (V, T, P, S)$ e $w \in T^*$, determinare se $S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \Rightarrow w$
  (se esiste un albero sintattico G con radice S e prodotto w)
- La costruzione dell'automa corrispondente a G produce un PDA non deterministico
- Per alcune G sappiamo che non è possibile trovare un DPDA

PARSING TOP-DOWN

Il parser cerca di ottenere una derivazione a sinistra $S \Rightarrow^*_{lm}$ in cui al passo i il parser sa che:

$$S \Rightarrow^*_{lm} uA\beta$$

e deve stabilire se

$$uA\beta \Rightarrow^*_{lm} w$$

- Se u non è prefisso di w, allora il parser rifiuta w
- Se $w = uav$ allora il parser deve scegliere una produzione per riscrivere $A \to \alpha_1 | \dots | \alpha_n$

scelgo lui

$A \to aA | b$

$A \to aB | aC$ non so chi scegliere

STRINGHE ANNULLABILI

Data $G=(V,T,P,S)$, diciamo che $\alpha \in (V \cup T)^*$ è annullabile ($NULL(\alpha)$), se e solo se $\alpha \Rightarrow^*_G \varepsilon$, ovvero se α può essere riscritta nella stringa vuota

① Se $NULL(X_1), \dots, NULL(X_n)$, allora $NULL(X_1 \dots X_n)$
② Se $\exists$ produzione $A \to \alpha \in P$ e $NULL(\alpha)$, allora $NULL(A)$

FIRST DI UNA STRINGA

Data $G=(V,T,P,S)$ e $\alpha \in (V \cup T)^*$, indichiamo con $FIRST(\alpha)$ gli inizi di α, ovvero l'insieme di simboli terminali che possono trovarsi all'inizio delle stringhe derivate da α

$$FIRST(\alpha) \overset{def}{=} \{ a \in T \mid \alpha \Rightarrow^*_G a\beta \}$$

ALGORITMO PER IL CALCOLO DI $FIRST(\alpha)$

- $FIRST(\varepsilon) = \emptyset$
- $FIRST(a) = \{a\}$
- $FIRST(A) = \bigcup_{A \to \alpha} FIRST(\alpha)$
- $FIRST(X\alpha) = \begin{cases} FIRST(X) \cup FIRST(\alpha) & \text{se } NULL(X) \\ FIRST(X) & \text{altrimenti} \end{cases}$

esempio

$S \to Ac \mid Ba$
$A \to \varepsilon \mid a$
$B \to b$
$C \to a \mid Cb$
$D \to \varepsilon \mid d \mid Db$

$NULL(A)$ e $NULL(D)$

① $FIRST(S) = FIRST(Ac) \cup FIRST(Ba) = \underbrace{FIRST(A) \cup FIRST(c)}_{\text{perché è annullabile}} \cup FIRST(B) = \{a\} \cup \{c\} \cup \{b\}$

② $FIRST(A) = FIRST(\varepsilon) \cup FIRST(a) = \emptyset \cup \{a\} = \{a\}$
③ $FIRST(B) = FIRST(b) = \{b\}$
④ $FIRST(C) = FIRST(a) \cup FIRST(Cb) = \{a\} \cup FIRST(C) = \{a\}$ togliamo perché siamo interessati al più piccolo insieme
⑤ $FIRST(D) = FIRST(\varepsilon) \cup FIRST(d) \cup FIRST(Db) = \emptyset \cup \{d\} \cup \cancel{FIRST(D)} \cup FIRST(b) = \{d\} \cup \{b\}$

FOLLOW DI UNA VARIABILE

Data $G = (V, T, P, S)$ e $A \in V$, indichiamo con FOLLOW(A) i seguiti di A, ovvero l'insieme dei simboli terminali che possono seguire A in una forma sequenziale

$$\text{FOLLOW}(A) \overset{def}{=} \{ a \in T \mid S \Rightarrow_G^* \alpha A a \beta \}$$

Per convenzione aggiungeremo la sentinella \$ ai seguiti del simbolo iniziale S in maniera tale che il parser possa riconoscere il fine stringa

ALGORITMO PER IL CALCOLO DI FOLLOW

FASE 1

Annotiamo le relazioni di appartenenza ed inclusione

- Annotare $\$ \in \text{FOLLOW}(S)$
- Ripetere i passi seguenti $\forall$ produzione e $\forall$ variabile nel corpo di queste:
  1. Se $A \to \alpha B \beta$, allora annotare $\text{FIRST}(\beta) \subseteq \text{FOLLOW}(B)$
  2. Se $A \to \alpha B \beta$ e $\text{NULL}(\beta)$, allora annotare $\text{FOLLOW}(A) \subseteq \text{FOLLOW}(B)$
  (se $A \to \alpha B$, allora annotare $\text{FOLLOW}(A) \subseteq \text{FOLLOW}(B)$)

FASE 2

Si determinino i seguiti propagando i simboli terminali (e \$) rispettando l'ordine delle inclusioni insiemistiche $\subseteq$ che sono state annotate

esempio

$S \to Ac \mid Ba$ NULL(A) e NULL(D)
$A \to \varepsilon \mid a$
$B \to b$
$C \to a \mid Cb$
$D \to \varepsilon \mid d \mid Db$

$\$ \in \text{FOLLOW}(S)$
$c \in \text{FOLLOW}(A)$
$a \in \text{FOLLOW}(B)$
$b \in \text{FOLLOW}(C)$
$b \in \text{FOLLOW}(D)$

	FOLLOW
S	$\{ \$ \}$
A	$\{ c \}$
B	$\{ a \}$
C	$\{ b \}$
D	$\{ b \}$

esempio

$E \to TE'$
$E' \to +TE' \mid \varepsilon$ NULL(E')
$T \to FT'$
$T' \to *FT' \mid \varepsilon$ NULL(T')
$F \to (E) \mid id$

$\$ \in$ FOLLOW(E)
FIRST$(E') \in$ FOLLOW(T)
FOLLOW$(E) \subseteq$ FOLLOW(T)
FOLLOW$(E) \subseteq$ FOLLOW(E')
FOLLOW$(E') \subseteq$ FOLLOW(T)
$*$ $\in$ FOLLOW(F)
FOLLOW$(T') \subseteq$ FOLLOW(F)
FOLLOW$(T) \subseteq$ FOLLOW(F)
FOLLOW$(T) \subseteq$ FOLLOW(T')

FOLLOW$(T') \subseteq$ FOLLOW(F)

FIRST$( )) \in$ FOLLOW(E) $\longrightarrow$ $) \in$ FOLLOW(E)

X	FOLLOW(X)
E	\$,)
E'	\$,)
T	\$, + ,)
T'	\$, + ,)
F	\$, + , * ,)

INSIEMI GUIDA

Data $G = (V, T, P, S)$ e prod $A \to \alpha$, indichiamo con GUIDA$(A \to \alpha)$ l'insieme guida di $A \to \alpha$:

$$\text{GUIDA}(A \to \alpha) \stackrel{def}{=} \begin{cases} \text{FIRST}(\alpha) \cup \text{FOLLOW}(A) & \text{se NULL}(\alpha) \\ \text{FIRST}(\alpha) & \text{altrimenti} \end{cases}$$

GRAMMATICHE LL(1)

Diciamo che una grammatica $G = (V, T, P, S)$ è LL(1) se per ogni coppia di produzioni distinte $A \to \alpha$ e $A \to \beta$ in P abbiamo che

$$\text{GUIDA}(A \to \alpha) \cap \text{GUIDA}(A \to \beta) = \emptyset$$

L $\to$ la stringa viene analizzata da sx a dx
L $\to$ derivazione canonica a sx (LEFTMOST)
1 $\to$ il parser utilizza 1 simbolo terminale della stringa per scegliere la produzione