nguaggi Formali e Traduttori

4.1 Parsing top-down e grammatiche LL(1)

- Sommario
- Strategia per il parsing top-down
- Stringhe annullabili (NULL)
- Esempi di stringhe annullabili
- Inizi di una stringa (FIRST)
- Come calcolare FIRST
- Esempi di calcolo di FIRST
- Seguiti di una variabile (FOLLOW)
- Come calcolare FOLLOW
- Esempi di calcolo di FOLLOW
- Insiemi guida
- Grammatiche LL(1)
- Esempio: espressioni aritmetiche
- Esercizi

È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Sommario

Problema

ullet Data una grammatica G=(V,T,P,S) e una stringa $w\in T^*$, determinare se

$$S\Rightarrow lpha_1\Rightarrow lpha_2\Rightarrow \cdots\Rightarrow w$$

o, equivalentemente, se esiste un albero sintattico di G con radice S e prodotto w.

- ullet La costruzione dell'automa corrispondente a $oldsymbol{G}$ produce un PDA non deterministico.
- ullet Per alcune G sappiamo che non è possibile trovare un DPDA.

In questa lezione

- Identifichiamo una famiglia di grammatiche libere per le quali è possibile costruire riconoscitori (parser) deterministici, cioè che non fanno uso di <u>backtracking</u>.
- Questi parser sono detti **top-down** perché costruiscono l'albero sintattico di w dalla radice (top) verso le foglie (down) o, equivalentemente, cercano una <u>derivazione sinistra</u> per w.

Strategia per il parsing top-down

Data una grammatica G = (V, T, P, S) e una stringa $w \in T^*$, il parser cerca di ottenere una derivazione a sinistra $S \Rightarrow_{lm}^* w$ in cui, al passo i, il parser sa che

$$S \Rightarrow_{lm}^* uAeta^{'}$$

e deve stabilire se

$$uAeta \Rightarrow_{lm}^* w$$

Ci sono due casi da considerare:

- lacksquare Se $oldsymbol{u}$ non $oldsymbol{\dot{e}}$ prefisso di $oldsymbol{w}$, allora il parser **rifiuta oldsymbol{w}**.
- $igcup_{ullet}$ Se $oldsymbol{w}=oldsymbol{u}aoldsymbol{v}_{ullet}$ allora il parser deve **scegliere** una produzione per riscrivere $oldsymbol{A}$

$$A olpha_1 |\cdots|lpha_n$$

e per farlo può usare a come "guida", a patto che tale simbolo identifichi univocamente l' α_i tale che $\alpha_i\beta \Rightarrow_{lm}^* av$.

Per ogni produzione $A \to \alpha_i$ occorre saper calcolare gli insiemi di simboli terminali che possono iniziare le stringhe derivate da $\alpha_i\beta$ e richiedere che tali insiemi siano disgiunti.

Stringhe annullabili (NULL)

Definizione

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

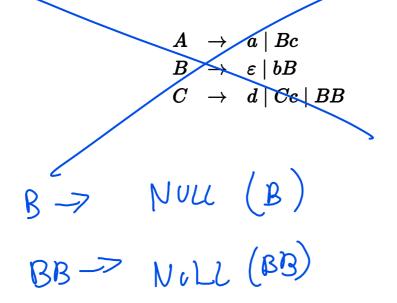
Come determinare se una stringa è annullabile

- (1) Se $\operatorname{NULL}(X_1), \ldots, \operatorname{NULL}(X_n)$, allora $\operatorname{NULL}(X_1 \cdots X_n)$.
- (2) Se esiste una produzione $A o lpha \in P$ e $\mathrm{NULL}(lpha)$, allora $\mathrm{NULL}(A)$.

Note

- Come caso particolare di (1) quando n=0 abbiamo $\text{NULL}(\varepsilon)$.
- Combinando (1) e (2) abbiamo che $A o arepsilon \in P$ implica $\mathtt{NULL}(A)$.
- Una stringa che contiene simboli terminali non è mai annullabile.

Esempi di stringhe annullabili



Esempi di stringhe annullabili

$$egin{array}{lll} A &
ightarrow & a \mid Bc \ B &
ightarrow & arepsilon \mid bB \ C &
ightarrow & d \mid Cc \mid BB \end{array}$$

- Da $\mathtt{NULL}(arepsilon)$ e dalla produzione B o arepsilon deduciamo $\mathtt{NULL}(B)$.
- Da $\mathtt{NULL}(B)$ e dalla produzione C o BB deduciamo $\mathtt{NULL}(C)$.
- Da NULL(B) e NULL(C) deduciamo NULL(BC).
- Da $\neg \text{NULL}(a)$ e $\neg \text{NULL}(Bc)$ deduciamo $\neg \text{NULL}(A)$.

Inizi di una stringa (FIRST)

Definizione

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

$$ext{FIRST}(lpha) \stackrel{\mathsf{def}}{=} \{a \in T \mid lpha \Rightarrow_G^* aeta\}$$

Attenzione

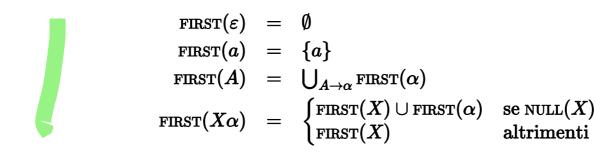
Il libro di testo usa un'unica funzione **FIRST** libro che <u>combina</u> **NULL** e **FIRST** così:

$$ext{FIRST}_{libro}(lpha) = egin{cases} ext{FIRST}(lpha) \cup \{arepsilon\} & ext{se NULL}(lpha) \ ext{FIRST}(lpha) & ext{altrimenti} \end{cases}$$

In pratica, l'approccio seguito dal libro ammette il simbolo speciale ε tra gli inizi di α per indicare il fatto che α è annullabile. Noi abbiamo definito un predicato $\text{NULL}(\alpha)$ apposito mentre $\text{FIRST}(\alpha)$ contiene solo simboli terminali.

Come calcolare FIRST

È possibile calcolare $\mathbf{FIRST}(\alpha)$ per induzione su α , usando le seguenti regole:



Attenzione

Applicando le regole qui sopra, può capitare di arrivare a equazioni della forma

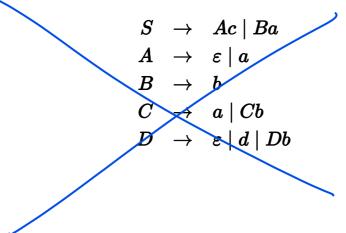
$$ext{First}(A) = ext{First}(A) \cup \mathcal{S}$$

dove ${oldsymbol{\mathcal{S}}}$ è un insieme di terminali. Questa equazione si può semplificare a

$$ext{first}(A) = \mathcal{S}$$

in quanto siamo interessati a ottenere <u>il più piccolo insieme</u> di terminali con la proprietà descritta nella slide precedente.

Esempi di calcolo di FIRST



Esempi di calcolo di FIRST

```
egin{array}{lll} S & 
ightarrow & Ac \mid Ba \ A & 
ightarrow & arepsilon \mid a \ B & 
ightarrow & b \ C & 
ightarrow & a \mid Cb \ D & 
ightarrow & arepsilon \mid d \mid Db \end{array}
```

Variabili annullabili

- NULL(A)
- $\mathrm{NULL}(D)$

Calcolo di FIRST di tutte le variabili

- $FIRST(B) = FIRST(b) = \{b\}$
- $\operatorname{FIRST}(A) = \operatorname{FIRST}(\varepsilon) \cup \operatorname{FIRST}(a) = \{a\}$
- $\bullet \ \ \operatorname{FIRST}(S) = \operatorname{FIRST}(Ac) \cup \operatorname{FIRST}(Ba) = \operatorname{FIRST}(A) \cup \operatorname{FIRST}(c) \cup \operatorname{FIRST}(B) = \{a,b,c\}$
- $\operatorname{FIRST}(C) = \operatorname{FIRST}(a) \cup \operatorname{FIRST}(Cb) = \{a\} \cup \operatorname{FIRST}(C) = \{a\}$
- $\bullet \ \ \operatorname{first}(D) = \operatorname{first}(\varepsilon) \cup \operatorname{first}(d) \cup \operatorname{first}(Db) = \{d\} \cup \operatorname{first}(D) \cup \operatorname{first}(b) = \{b,d\}$

Seguiti di una variabile (FOLLOW)

Definizione

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



$$ext{FOLLOW}(A) \stackrel{\mathsf{def}}{=} \{a \in T \mid S \Rightarrow_G^* lpha Aaeta \}$$

Attenzione

- ullet Per convenzione aggiungeremo una sentinella ullet ai seguiti del simbolo iniziale S.
- In questo modo il parser può capire quando è arrivato alla fine della stringa da riconoscere.

Come calcolare FOLLOW

Il calcolo di FOLLOW si effettua in due fasi.

Fase 1

In questa fase si annotano relazioni di appartenenza ed inclusione insiemistica secondo il seguente algoritmo:

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

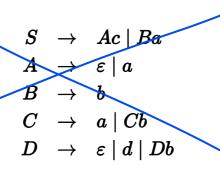
Caso particolare di (2): se $A \to \alpha B$, allora annotare $FOLLOW(A) \subseteq FOLLOW(B)$.

Fase 2

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

Per grammatiche complesse può essere utile fare una tabella con due colonne, l'elenco di tutte le variabili nella prima ed i seguiti corrispondenti alle variabili nella seconda.

Esempi di calcolo di FOLLOW



Esempi di calcolo di FOLLOW

$$egin{array}{lll} S &
ightarrow & Ac \mid Ba \ A &
ightarrow & arepsilon \mid a \ B &
ightarrow & b \ C &
ightarrow & a \mid Cb \ D &
ightarrow & arepsilon \mid d \mid Db \end{array}$$

Fase 1

- $\$ \in \text{FOLLOW}(S)$
- $\operatorname{FIRST}(c) \subseteq \operatorname{FOLLOW}(A)$
- $FIRST(a) \subseteq FOLLOW(B)$
- $FIRST(b) \subseteq FOLLOW(C)$
- $\operatorname{first}(b) \subseteq \operatorname{Follow}(D)$

Fase 2

X	$\operatorname{FOLLOW}(X)$
S	{\$ }
$oldsymbol{A}$	$\{c\}$
B	$\{a\}$
C	$\{b\}$
D	$\{b\}$

Insiemi guida

Definizione

Data una grammatica G=(V,T,P,S) e una produzione A olpha, indichiamo con GUIDA(A olpha) l'insieme guida di A olpha, ovvero l'insieme

$$ext{GUIDA}(A
ightarrow lpha) \stackrel{\mathsf{def}}{=} egin{cases} ext{FIRST}(lpha) \cup ext{FOLLOW}(A) & ext{se NULL}(lpha) \ ext{FIRST}(lpha) & ext{altrimenti} \end{cases}$$

Intuizione

Un parser predittivo che sceglie di riscrivere la variabile A usando la produzione $A \to \alpha$ si aspetta di leggere nella stringa di input uno dei simboli nell'insieme guida di $A \to \alpha$.

Sono due i casi da considerare:

- 1. Il simbolo è uno degli inizi di α , oppure
- 2. α è annullabile ed il simbolo è uno dei seguiti di A.

Grammatiche LL(1)

Definizione

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

$$\mathtt{GUIDA}(A olpha)\cap\mathtt{GUIDA}(A oeta)=\emptyset$$

Intuizione

Noto il simbolo da riscrivere A, note le produzioni $A \to \beta_1 \mid \cdots \mid \beta_n$ e noto il prossimo simbolo terminale a nella stringa da riconoscere, in una grammatica $\mathrm{LL}(1)$ esiste al massimo una produzione "giusta" tale che $a \in \mathrm{GUIDA}(A \to \beta_i)$ dunque il parser predittivo <u>identifica univocamente</u> la produzione $A \to \beta_i$ a partire da a.

Cosa c'è nel nome LL(1)

- L \rightarrow la stringa in input viene analizzata <u>da sinistra (left) a destra</u>;
- L \rightarrow il parser cerca di costruire una derivazione canonica sinistra (leftmost);
- $1 \rightarrow \text{il parser usa } \underline{\text{un solo simbolo terminale}}$ della stringa per scegliere la produzione.

Esempio: espressioni aritmetiche

$$egin{array}{lcl} E &
ightarrow TE' \ E' &
ightarrow +TE' ig| arepsilon & ext{NULL}(E') \ T &
ightarrow FT' \ T' &
ightarrow *FT' ig| arepsilon & ext{NULL}(T') \ F &
ightarrow & (E) ig| ext{id} \end{array}$$

- $\$ \in \text{Follow}(E)$
- $\{+\} = \operatorname{FRST}(E') \subseteq \operatorname{FOLLOW}(T)$
- $follow(E) \subseteq follow(T)$
- FOLLOW $(E) \subseteq \text{FOLLOW}(E')$
- $\operatorname{Follow}(E') \subseteq \operatorname{Follow}(T)$
- $\{*\} = \operatorname{FIRST}(T') \subseteq \operatorname{FOLLOW}(F)$
- FOLLOW(T) \subseteq FOLLOW(F)
- FOLLOW $(T) \subseteq \text{FOLLOW}(T')$
- $\text{FOLLOW}(T') \subseteq \text{FOLLOW}(F)$
- $\{)\} = FIRST()) \subseteq FOLLOW(E)$

Esempio: espressioni aritmetiche

 $egin{aligned} & ext{FIRST}(E) = ext{FIRST}(T) = \{ ext{(,id}\} \ & ext{FIRST}(T) = ext{FIRST}(F) = \{ ext{(,id}\} \ & ext{FIRST}(T') = \{ ext{*}\} \ & ext{FIRST}(F) = \{ ext{(,id}\} \end{aligned}$

- $\$ \in \text{FOLLOW}(E)$
- $\{+\} = \operatorname{FRST}(E') \subseteq \operatorname{FOLLOW}(T)$
- $follow(E) \subseteq follow(T)$
- $FOLLOW(E) \subseteq FOLLOW(E')$
- $\text{FOLLOW}(E') \subseteq \text{FOLLOW}(T)$
- $\{*\} = \operatorname{FIRST}(T') \subseteq \operatorname{FOLLOW}(F)$
- $FOLLOW(T) \subseteq FOLLOW(F)$
- FOLLOW $(T) \subseteq \text{FOLLOW}(T')$
- FOLLOW $(T') \subseteq \text{FOLLOW}(F)$
- $\{\}$ = First()) \subseteq Follow(E)

Esempio: espressioni aritmetiche

$$egin{array}{lcl} E &
ightarrow & TE' \ E' &
ightarrow & +TE' \mid arepsilon & ext{NULL}(E') \ T &
ightarrow & FT' \ T' &
ightarrow & *FT' \mid arepsilon & ext{NULL}(T') \ F &
ightarrow & (E) \mid ext{id} \end{array}$$

- $\$ \in \text{FOLLOW}(E)$
- $\{+\} = \operatorname{FIRST}(E') \subseteq \operatorname{FOLLOW}(T)$
- $follow(E) \subseteq follow(T)$
- $FOLLOW(E) \subseteq FOLLOW(E')$
- $\text{FOLLOW}(E') \subseteq \text{FOLLOW}(T)$
- $\{*\} = FRST(T') \subseteq FOLLOW(F)$
- $FOLLOW(T) \subseteq FOLLOW(F)$
- $follow(T) \subseteq follow(T')$
- $\text{FOLLOW}(T') \subseteq \text{FOLLOW}(F)$
- $\{)\} = FIRST()) \subseteq FOLLOW(E)$

$$egin{aligned} & ext{FIRST}(E) = ext{FIRST}(T) = \{ ext{(,id}\} \ & ext{FIRST}(T) = ext{FIRST}(F) = \{ ext{(,id}\} \ & ext{FIRST}(T') = \{*\} \ & ext{FIRST}(F) = \{ ext{(,id}\} \end{aligned}$$

X	$\operatorname{FOLLOW}(X)$
$oldsymbol{E}$	\$,)
E'	\$,)
T	\$,),+
T'	\$,),+
$oldsymbol{F}$	\$,),+,*

Esercizi

- 1. Calcolare gli insiemi guida della grammatica nella slide 14. La grammatica è LL(1)?
- 2. Calcolare gli insiemi guida della seguente grammatica e determinare se è LL(1).

$$egin{array}{lcl} A &
ightarrow & BC \mid D \ B &
ightarrow & arepsilon \mid a \ C &
ightarrow & b \mid cCc \ D &
ightarrow & arepsilon \mid CD \end{array}$$

3. Ripetere l'esercizio precedente per la grammatica

$$egin{array}{lll} S &
ightarrow & ext{if E then SS' fi $|$ skip} \ S' &
ightarrow & ext{else S} | arepsilon \ E &
ightarrow & ext{true} | ext{false} \end{array}$$

in cui S, S' ed E sono variabili e \mathbf{if} , \mathbf{then} , ... sono terminali.

4. Ripetere l'esercizio precedente dopo aver rimosso il terminale **fi** dalla grammatica.