Proprietà dei linguaggi CF

a.a. 2020-2021

Corso di Fondamenti di Informatica - 1 modulo

Corso di Laurea in Informatica

Università di Roma "Tor Vergata"

Prof. Giorgio Gambosi

Chiusura dei linguaggi CF: intersezione

Il linguaggio $L = \{a^n b^n c^n \mid n \geq 1\}$ non è context free.

Del resto, $L_1 = \{a^n b^n c^m \mid n, m \geq 1\}$ e $L_2 = \{a^m b^n c^n \mid n, m \geq 1\}$ sono non contestuali

Ma $L = L_1 \cap L_2$, da cui deriva che la classe dei linguaggi CF non è chiusa rispetto all'intersezione

Chiusura dei linguaggi CF: unione

Dati due linguaggi context free $L_1 \subseteq \Sigma_1^*$ e $L_2 \subseteq \Sigma_2^*$, siano $\mathcal{G}_1 = \langle \Sigma_1, V_{N1}, P_1, S_1 \rangle$ e $\mathcal{G}_2 = \langle \Sigma_2, V_{N2}, P_2, S_2 \rangle$ due grammatiche di tipo 2 tali che $L_1 = L(\mathcal{G}_1)$ e $L_2 = L(\mathcal{G}_2)$.

Il linguaggio $L = L_1 \cup L_2$ potrà allora essere generato dalla grammatica di tipo 2 $\mathcal{G} = \langle \Sigma_1 \cup \Sigma_2, V_{N1} \cup V_{N2} \cup \{S\}, P, S \rangle$, dove $P = P_1 \cup P_2 \cup \{S \longrightarrow S_1 \mid S_2\}$.

Chiusura dei linguaggi CF: concatenazione

Dati due linguaggi context free $L_1 \subseteq \Sigma_1^*$ e $L_2 \subseteq \Sigma_2^*$, siano $\mathcal{G}_1 = \langle \Sigma_1, V_{N1}, P_1, S_1 \rangle$ e $\mathcal{G}_2 = \langle \Sigma_2, V_{N2}, P_2, S_2 \rangle$ due grammatiche di tipo 2 tali che $L_1 = L(\mathcal{G}_1)$ e $L_2 = L(\mathcal{G}_2)$.

Mostriamo che il linguaggio $L = L_1 \circ L_2$ è generato dalla grammatica di tipo 2 definita come $\mathcal{G} = \langle \Sigma_1 \cup \Sigma_2, V_{N1} \cup V_{N2} \cup \{S\}, P, S \rangle$, dove $P = P_1 \cup P_2 \cup \{S \longrightarrow S_1 S_2\}$.

Chiusura dei linguaggi CF: iterazione

Dato un linguaggio context free $L \subseteq \Sigma^*$, sia $\mathcal{G} = \langle \Sigma, V_N, P, S \rangle$ una grammatica di tipo 2 tale che $L = L(\mathcal{G})$.

Il linguaggio $L' = L^*$ è allora generato dalla grammatica di tipo 2 $\mathcal{G}' = \langle \Sigma, V_N \cup \{S'\}, P', S' \rangle$, dove $P' = P \cup \{S' \longrightarrow SS' \mid \varepsilon\}$.

Chiusura dei linguaggi CF: complemento

La classe dei linguaggi CF non è chiusa rispetto al complemento.

Infatti, se cosìfosse, avremmo che dati due qualunque linguaggi CF L_1, L_2, il linguaggio $L = \overline{\overline{L_1} \cup \overline{L_2}}$ sarebbe CF anch'esso. Ma $L = L_1 \cap L_2$ e quindi ne risulterebbe la chiusura rispetto all'intersezione, che non sussiste.

Decidibilità di predicati su linguaggi CF

Data una grammatica $\mathcal{G}$ di tipo 2 è decidibile stabilire se $L(G) = \emptyset$.

Assumiamo che $\mathcal{G} = \langle V_T, V_N, P, S \rangle$ sia in CNF e poniamo $n = |V_N|$.

Per il pumping lemma, se esiste una stringa $z = uvwxy \in L(\mathcal{G})$ con $|z| > 2^n$, allora esiste una stringa $z' = uwy \in L(\mathcal{G})$ con $|z'| \leq 2^n$. Quindi, se il linguaggio non è vuoto, esiste una stringa in esso di lunghezza al più 2^n

Decidibilità di predicati su linguaggi CF

In una grammatica in CNF ogni applicazione di una produzione o incrementa di uno la lunghezza della forma di frase (se la produzione è del tipo $A \longrightarrow BC$) o sostituisce un terminale a un non terminale (se è del tipo $A \longrightarrow a$). Quindi, una stringa di lunghezza k è generata da una derivazione di lunghezza $2k - 1$

Per verificare se esiste una stringa di lunghezza al più 2^n generabile, è sufficiente considerare tutte le derivazioni di lunghezza al più $2^{n+1} - 1 < |P|\, 2^{|V_N|+1}$.

Decidibilità di predicati su linguaggi CF

Un metodo più efficiente consiste nel portare la grammatica in forma ridotta, verificando se esistono simboli fecondi. Condizione necessaria e sufficiente affinchè il linguaggio sia vuoto è che la grammatica non abbia simboli fecondi.

Decidibilità di predicati su linguaggi CF

Data una grammatica $\mathcal{G}$ di tipo 2 è decidibile stabilire se $L(G)$ è infinito.

Assumiamo che $\mathcal{G} = \langle V_T, V_N, P, S \rangle$ sia in CNF e poniamo $n = |V_N|$.

Per il pumping lemma, se esiste una stringa $z = uvwxy \in L(\mathcal{G})$ con $2^n < |z| \leq 2^{2n}$, allora esistono infinite stringhe $z_i = uv^iwx^iy \in L(\mathcal{G})$, con $i \geq 0$.

Quindi, se il linguaggio è infinito, esiste una stringa in esso di lunghezza compresa tra $2^n + 1$ e 2^{2n}

Decidibilità di predicati su linguaggi CF

È possibile allora considerare tutte le derivazioni di lunghezza al più $2^{2(n+1)} - 1 < |P| \left(2^{(2|V_N|+1)}\right)$, e verificare se qualcuna di esse dà origine ad una stringa di terminali.

Metodo più efficiente: verificare se il grafo $G = (N, A)$ è ciclico. N corrisponde ai non terminali della grammatica, assunta in CNF. Per ogni produzione $B \longrightarrow CD$ il grafo contiene gli archi (B, C) e (B, D)

Ambiguità

Una grammatica $\mathcal{G}$ si dice ambigua se esiste una stringa x in $L(\mathcal{G})$ derivabile con due diversi alberi sintattici.

L'albero sintattico di una stringa corrisponde in qualche modo al significato della stringa stessa, quindi l'univocità di questo albero è importante per comprendere senza ambiguità tale significato

Ambiguità

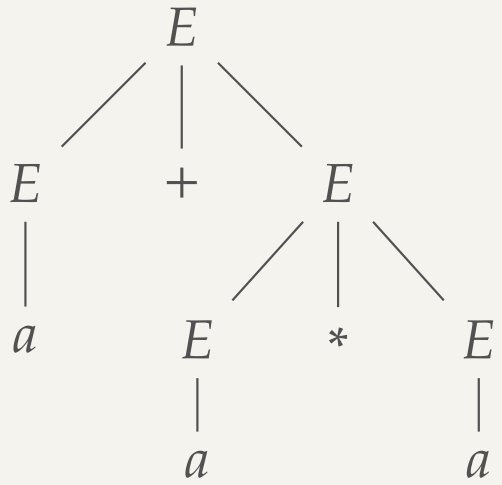
Si consideri la grammatica

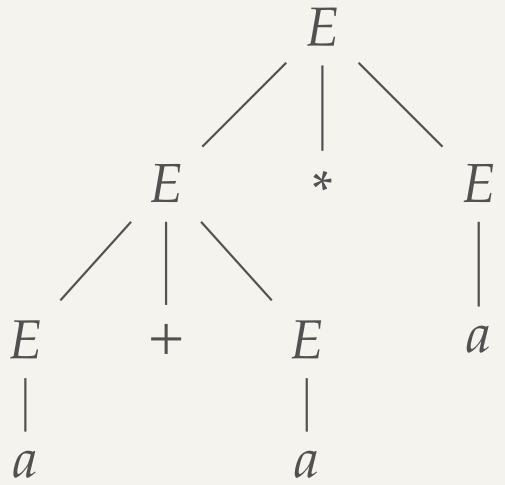
$$E \longrightarrow E + E \mid E - E \mid E * E \mid E/E \mid (E) \mid a.$$

Essa genera tutte le espressioni aritmetiche sulla variabile a, ma come si vede facilmente la stessa espressione può essere derivata con alberi di derivazione diversi.

Ambiguità

Ad esempio la stringa $a + a * a$ può venire derivata mediante due diversi alberi.





Ambiguità

Si consideri la grammatica

$$E \longrightarrow E + E \mid E - E \mid E * E \mid E/E \mid (E) \mid a.$$

Essa genera tutte le espressioni aritmetiche sulla variabile a, ma come si vede facilmente la stessa espressione può essere derivata con alberi di derivazione diversi.

Ambiguità

Eliminazione dell'ambiguità:

- Introduzione di parentesi
- Precedenza tra operatori

Ambiguità

Parentesi:

$$E \longrightarrow (E + E) \mid (E - E) \mid (E * E) \mid (E/E) \mid (E) \mid a.$$

I due diversi alberi di derivazione che davano origine alla stessa stringa, danno ora origine alle due stringhe

$$(a + (a * a))$$
$$((a + a) * a).$$

Ambiguità

Precedenza tra operatori:

$$
\begin{aligned}
E &\longrightarrow E + T \mid E - T \mid T \\
T &\longrightarrow T * F \mid T/F \mid F \\
F &\longrightarrow (E) \mid a
\end{aligned}
$$

La grammatica rappresenta nella sua struttura le relazioni di precedenza definite tra gli operatori (nell'ordine non decrescente +,-,*,/) e in tal modo consente di utilizzare le parentesi soltanto quando strettamente necessario.

Ambiguità

Riconoscimento: Data una grammatica $\mathcal{G}$ non contestuale, $\mathcal{G}$ è ambigua?

Il problema è indecidibile nel caso di CFG: non esiste quindi nessun algoritmo di decisione che, data una CFG, restituisca T se la grammatica è ambigua e F altrimenti.

Riduzioni

Indecidibilità dimostrata mediante riduzione da un altro problema di decisione $\mathcal{P}$, che si sa essere indecidibile.

Schema generale di dimostrazione:

- si vuole mostrare che il problema $\mathcal{P}_1$ è indecidibile
- si individua un altro problema $\mathcal{P}_0$ che si sa essere indecidibile
- si definisce un algoritmo $\mathcal{A}$ che trasforma ogni istanza I_0 di $\mathcal{P}_0$ in una istanza $I_1 = \mathcal{A}(I_0)$ di $\mathcal{P}_1$
- si mostra che l'istanza I_1 è positiva per $\mathcal{P}_1$ se e solo I_0 è positiva per $\mathcal{P}_0$
- si conclude che $\mathcal{P}_1$ è indecidibile: se così non fosse avremmo una algoritmo che decide $\mathcal{P}_0$, in quanto potremmo trasformare, per mezzo di $\mathcal{A}$, ogni sua istanza in una istanza corrispondente di $\mathcal{P}_1$ che potremmo, per ipotesi, risolvere

Ambiguità

Nel nostro caso:

- $\mathcal{P}_1$ è il problema di determinare, data una grammatica CF (istanza del problema), se essa è ambigua
- $\mathcal{P}_0$ è PCP (Problema delle Corrispondenze di Post):
  - data una istanza del problema, composta da:
    - un alfabeto Σ
    - due sequenze di k parole $X = x_1, \ldots, x_k$ e $Y = y_1, \ldots, y_k$ costruite su Σ
  - esiste una sequenza di $m \geq 1$ interi $i_1, i_2, \ldots, i_m$ in $[1, \ldots, k]$ tale che risulti

$$x_{i_1} x_{i_2} \ldots x_{i_m} = y_{i_1} y_{i_2} \ldots y_{i_m}?$$

Esempio di PCP

- Consideriamo le due sequenze $1, 10111, 10$ e $111, 10, 0$ costruite sull'alfabeto $\{0, 1\}$
- si può verificare che la sequenza di interi $2, 1, 1, 3$ costituisce una soluzione alla istanza di PCP considerata.
- infatti, si ottiene in un caso la sequenza
  $10111 \cdot 1 \cdot 1 \cdot 10 = 101111110$ e nell'altro la stessa sequenza
  $10 \cdot 111 \cdot 111 \cdot 0 = 101111110$

PCP è indecidibile (dimostrazione per riduzione dal Problema della fermata)

Riduzione

- Sia $A = x_1, \ldots, x_k$ e $B = y_1, \ldots, y_k$ una istanza (generica) di PCP su un alfabeto Σ
- Consideriamo
  - l'alfabeto $\Sigma \cup \{a_1, a_2, \ldots, a_k\}$, con $a_i \notin \Sigma$, $i = 1, \ldots, k$
  - il linguaggio $L' = L_A \cup L_B$ definito su Σ, in cui:
    - $L_A = \{x_{i_1} x_{i_2} \cdots x_{i_m} a_{i_m} a_{i_{m-1}} \cdots a_{i_1} \mid m \geq 1\}$
    - $L_B = \{y_{i_1} y_{i_2} \cdots y_{i_m} a_{i_m} a_{i_{m-1}} \cdots a_{i_1} \mid m \geq 1\}$.
  - la relativa grammatica CF

$$\mathcal{G}' = \langle \{S, S_A, S_B\}, \Sigma \cup \{a_1, \ldots, a_k\}, P, S \rangle,$$

con produzioni P, per $i = 1, \ldots, k$:

$$\begin{aligned} S &\longrightarrow S_A \mid S_B \\ S_A &\longrightarrow x_1 S_A a_1 \mid \cdots \mid x_k S_A a_k \mid x_1 a_1 \mid \cdots \mid x_k a_k \\ S_B &\longrightarrow y_1 S_B a_1 \mid \cdots \mid y_k S_B a_k \mid y_1 a_1 \mid \cdots \mid y_k a_k \end{aligned}$$

Equivalenza tra istanze

Se l'istanza (A, B) di PCP ha soluzione allora $\mathcal{G}'$ è ambigua.

- Sia $i_1, \ldots, i_m$ una soluzione di PCP, tale che quindi $x_{i_1} \cdots x_{i_m} a_{i_m} \cdots a_{i_1} = y_{i_1} \cdots x_{i_m} a_{i_m} \cdots a_{i_1} = \sigma$.
- La stringa σ appartiene a L' e ammette due distinti alberi sintattici, corrispondi il primo alla derivazione

$$S \Longrightarrow S_A \Longrightarrow x_{i_1} S_A a_{i_1} \Longrightarrow x_{i_1} x_{i_2} S_A a_{i_2} a_{i_1} \overset{*}{\Longrightarrow} x_{i_1} \cdots x_{i_m} a_{i_m} \cdots a_{i_1},$$

e il secondo alla derivazione

$$S \Longrightarrow S_B \Longrightarrow y_{i_1} S_B a_{i_1} \overset{*}{\Longrightarrow} y_{i_1} \cdots y_{i_m} a_{i_m} \cdots a_{i_1} = x_{i_1} \cdots x_{i_m} a_{i_m} \cdots a_{i_1}.$$

- $\mathcal{G}'$ risulta dunque ambigua

Equivalenza tra istanze

Se $\mathcal{G}'$ è ambigua allora l'istanza (A, B) di PCP ha soluzione.

- Sia z una stringa di L' che ammette due distinti alberi sintattici
- Per definizione di L', deve essere $z = wa_{i_m} \cdots a_{i_1}$ per un qualche $m \geq 1$
- Inoltre, per definizione di L', z deve appartenere ad almeno uno tra L_A e L_B: assumiamo, senza perdere generalità, che $z \in L_A$
- Allora, deve essere $w = x_{i_1} \cdots x_{i_m}$, e la produzione iniziale della derivazione deve essere $S \to S_A$
- Ma per definizione di $\mathcal{G}'$, l'altro modo di derivare z non può che prevedere come prima produzione $S \to S_B$, per cui $w = y_{i_1} \cdots y_{i_m}$
- Ne deriva che $i_1, \ldots, i_m$ è una soluzione dell'istanza (A, B) di PCP

Indecidibilità

- La trasformazione definita deriva quindi da una istanza di PCP una grammatica CF che è ambigua se e solo se l'istanza ha soluzione
- Se avessimo un algoritmo che determina se una grammatica CF è ambigua, allora potremmo determinare se una istanza di PCP ha soluzione
- Ma un algoritmo che determina se una istanza di PCP ha soluzione non esiste
- Quindi, non esiste un algoritmo che determina se una grammatica CF è ambigua

Ambiguità

Esistenza di grammatica equivalente non ambigua: Un linguaggio di tipo 2 si dice inerentemente ambiguo se tutte le grammatiche che lo generano sono ambigue.

Anche il problema dell'inerente ambiguità di un linguaggio è indecidibile.