Logica e linguaggi Context-Free

Logica per l'Informatica: approfondimento

Roberto Borelli

Università degli Studi di Udine

9 gennaio 2023

Outline

Collocazione dei linguaggi Context-Free

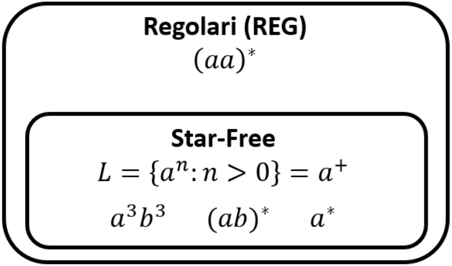
La gerarchia di chomsky

Star-Free

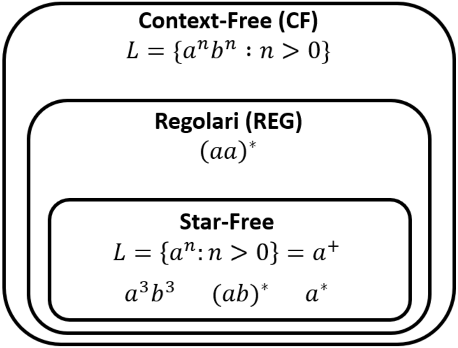
$$L = \{a^n : n > 0\} = a^+$$

$$a^3 b^3 \qquad (ab)^* \qquad a^*$$

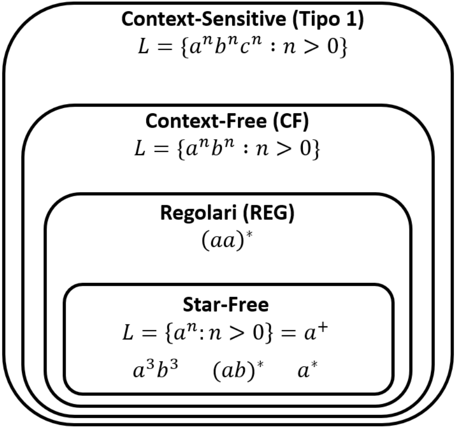
La gerarchia di chomsky



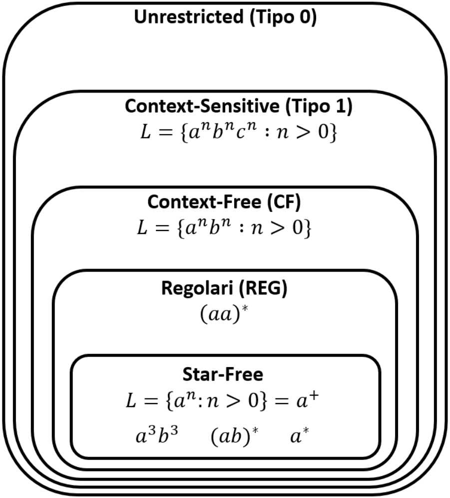
La gerarchia di chomsky



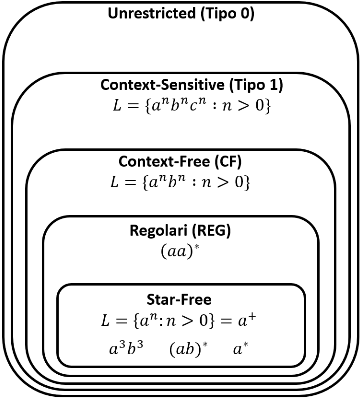
La gerarchia di chomsky



La gerarchia di chomsky

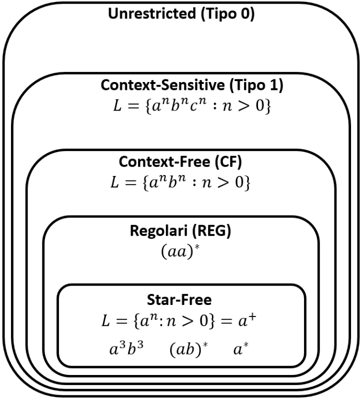


La gerarchia di chomsky



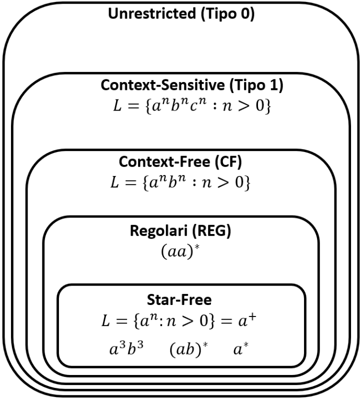
- **Tipo 0** Macchina di Turing
- **Tipo 1** Automa linear bounded
- **CF** Automa a pila
- **REG** Automa a stati finiti
- **STAR-FREE** Automa counter free

Caratterizzazione logica



- **STAR-FREE**
  Formule in $FO(=, <, (a(x))_{a \in \Sigma})$
- **REG**
  Formule in $MSO(=, <, (a(x))_{a \in \Sigma})$
  Teorema di Büchi, 1960

Caratterizzazione logica



- **STAR-FREE**
  Formule in $FO(=, <, (a(x))_{a \in \Sigma})$
- **REG**
  Formule in $MSO(=, <, (a(x))_{a \in \Sigma})$
  Teorema di Büchi, 1960
- **CF** ??

Potenza dei linguaggi CF

I seguenti linguaggi sono CF ma non REG:

Potenza dei linguaggi CF

I seguenti linguaggi sono CF ma non REG:

- $L_1 = \{a^n b^n : n > 0\}$ è CF
  $L_1 = \{ab, aabb, aaabbb, aaaabbbb, \dots\}$

Potenza dei linguaggi CF

I seguenti linguaggi sono CF ma non REG:

- $L_1 = \{a^n b^n : n > 0\}$ è CF
  $L_1 = \{ab, aabb, aaabbb, aaaabbbb, \ldots\}$
- Il linguaggio delle operazioni aritmetiche con parentesi bilanciate è CF
  $L_2 = \{\ldots, (2 * 2), ((5 + 4) * (5 * 8)) + 2, ((((4 + 5)))), \ldots\}$

Potenza dei linguaggi CF

I seguenti linguaggi sono CF ma non REG:

- $L_1 = \{a^n b^n : n > 0\}$ è CF
  $L_1 = \{ab, aabb, aaabbb, aaaabbbb, \dots\}$
- Il linguaggio delle operazioni aritmetiche con parentesi bilanciate è CF
  $L_2 = \{\dots, (2 * 2), ((5 + 4) * (5 * 8)) + 2, ((((4 + 5)))), \dots\}$
- Possiamo sommare...
  $L_3 = \{0^n 1^m 2^{n+m} : n + m > 0\}$
  $L_3 = \{02, 12, 001222, \dots, 000111122222222, \dots\}$

Limiti dei linguaggi CF

I seguenti linguaggi sono CONTEXT-SENSITIVE ma non CONTEXT-FREE:

Limiti dei linguaggi CF

I seguenti linguaggi sono CONTEXT-SENSITIVE ma non CONTEXT-FREE:

- $L_4 = \{a^n b^n c^n : n > 0\}$ non è CF
  (Si noti che $L_4 = \{a^n b^n c^m : n, m > 0\} \cap \{a^m b^n c^n : n, m > 0\}$)

Limiti dei linguaggi CF

I seguenti linguaggi sono CONTEXT-SENSITIVE ma non CONTEXT-FREE:

- $L_4 = \{a^n b^n c^n : n > 0\}$ non è CF
  (Si noti che $L_4 = \{a^n b^n c^m : n, m > 0\} \cap \{a^m b^n c^n : n, m > 0\}$)
- Il prodotto...
  $L_5 = \{0^n 1^m 2^{n*m} : n + m > 0\}$
  $L_5 = \{0, 1, 00122, \ldots, 000111122222222222, \ldots\}$

Grammatiche Context-Free

Grammatica: definizione

Una grammatica è una quadrupla G=<V,T,P,S>

- **V** insieme finito di simboli non terminali
- **T** insieme finito di simboli terminali
- **P** insieme finito di produzioni
- **S** simbolo iniziale (è un simbolo non terminale)

Grammatica: definizione

Una grammatica è una quadrupla G=<V,T,P,S>

- **V** insieme finito di simboli non terminali
- **T** insieme finito di simboli terminali
- **P** insieme finito di produzioni
- **S** simbolo iniziale (è un simbolo non terminale)

In una **grammatica CF** le produzioni sono del tipo $A \rightarrow \alpha$ dove:

- A è un simbolo non terminale
- $\alpha \in (V \cup T)^*$

Grammatica: produzioni e linguaggio generato

- Da $\alpha A \gamma$ **deriva immediatamente** $\alpha \beta \gamma$ se
  - In P c'è la produzione $A \to \beta$
  - α, γ appartengono a $(V \cup T)^*$

Scriveremo $\alpha A \gamma \Longrightarrow_1 \alpha \beta \gamma$

Grammatica: produzioni e linguaggio generato

- Da $\alpha A\gamma$ **deriva immediatamente** $\alpha\beta\gamma$ se
  - In P c'è la produzione $A \to \beta$
  - α, γ appartengono a $(V \cup T)^*$

  Scriveremo $\alpha A\gamma \Longrightarrow_1 \alpha\beta\gamma$
- Da α **deriva** β se esiste una sequenza (finita) di derivazioni immediate che mi permette di passare da α a β. Scriveremo $\alpha \Longrightarrow_* \beta$

Grammatica: produzioni e linguaggio generato

- Da $\alpha A\gamma$ **deriva immediatamente** $\alpha\beta\gamma$ se
  - In P c'è la produzione $A \rightarrow \beta$
  - α, γ appartengono a $(V \cup T)^*$

  Scriveremo $\alpha A\gamma \Longrightarrow_1 \alpha\beta\gamma$
- Da α **deriva** β se esiste una sequenza (finita) di derivazioni immediate che mi permette di passare da α a β. Scriveremo $\alpha \Longrightarrow_* \beta$
- Il **linguaggio generato** da una grammatica G, è l'insieme delle stringhe composte da soli simboli terminali tale per cui esiste una derivazione a partire dal simbolo iniziale. Ossia:
  $L(G) = \{w \in T^* : S \Longrightarrow_* w\}$

Grammatica: esempio 1

Il linguaggio $L_1 = \{a^n b^n : n > 0\}$ è generato dalla seguente grammatica:
$S \rightarrow ab \mid aSb$

Grammatica: esempio 1

Il linguaggio $L_1 = \{a^n b^n : n > 0\}$ è generato dalla seguente grammatica:
$S \to ab \mid aSb$

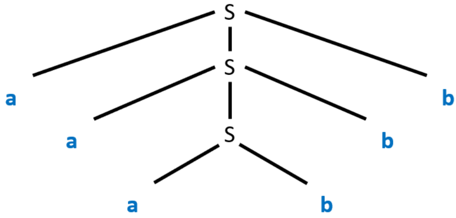
Ad esempio possiamo generare *aaabbb* in una derivazione di 3 passi:
$S \Longrightarrow_1 aSb \Longrightarrow_1 aaSbb \Longrightarrow_1 aaabbb$

Grammatica: esempio 1

Il linguaggio $L_1 = \{a^n b^n : n > 0\}$ è generato dalla seguente grammatica:
$S \rightarrow ab \mid aSb$

Ad esempio possiamo generare *aaabbb* in una derivazione di 3 passi:
$S \Longrightarrow_1 aSb \Longrightarrow_1 aaSbb \Longrightarrow_1 aaabbb$

La derivazione può anche essere rappresentata come un albero in cui nelle foglie leggiamo solo simboli terminali



Grammatica: esempio 2

Il linguaggio $L_3 = \{0^n 1^m 2^{n+m} : n + m > 0\}$ è generato dalla seguente grammatica:

Grammatica: esempio 2

Il linguaggio $L_3 = \{0^n 1^m 2^{n+m} : n + m > 0\}$ è generato dalla seguente grammatica:

$S \rightarrow 02 \mid 0S2 \mid B$

$B \rightarrow 12 \mid 1B2$

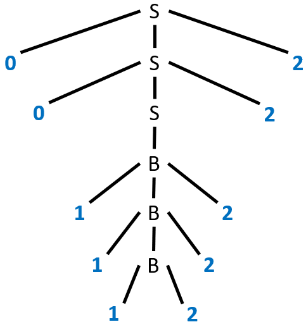
Grammatica: esempio 2

Il linguaggio $L_3 = \{0^n 1^m 2^{n+m} : n + m > 0\}$ è generato dalla seguente grammatica:

$S \rightarrow 02 \mid 0S2 \mid B$

$B \rightarrow 12 \mid 1B2$

L'albero di derivazione per 0011122222 è:



Normalizzazione di una grammatica CF

Lemma

Ogni linguaggio CF è generato da una grammatica CF che soddisfa:

- *Tutte le produzioni sono della forma:*

Normalizzazione di una grammatica CF

Lemma

Ogni linguaggio CF è generato da una grammatica CF che soddisfa:

- *Tutte le produzioni sono della forma:*
  - $S \to a$ *con* a *terminale*
  - $X \to a\gamma b$ *con* $\gamma \in (V \cup T)^*$ *e* a, b *terminali*

Normalizzazione di una grammatica CF

Lemma

Ogni linguaggio CF è generato da una grammatica CF che soddisfa:

- *Tutte le produzioni sono della forma:*
  - $S \to a$ *con* a *terminale*
  - $X \to a\gamma b$ *con* $\gamma \in (V \cup T)^*$ *e* a, b *terminali*
- *Se due produzioni non terminali hanno lo stesso* **pattern**, *allora hanno lo stesso termine sinistro*

Pattern

Data una produzione $X \to v_0 X_1 v_1 \ldots v_{n-1} X_n v_n$ dove $v_0, \ldots, v_n$ sono stringhe (anche vuote) di terminali
il suo **pattern** è la stringa $v_0 \# v_1 \# v_2 \# \ldots \# v_{n-1} \# v_n$
... ossia è la stringa dei terminali tra le variabili

Pattern

Data una produzione $X \to v_0 X_1 v_1 \dots v_{n-1} X_n v_n$ dove $v_0, \dots, v_n$ sono stringhe (anche vuote) di terminali
il suo **pattern** è la stringa $v_0 \# v_1 \# v_2 \# \dots \# v_{n-1} \# v_n$
... ossia è la stringa dei terminali tra le variabili

- Le produzioni
  $X \to aaXbYZv$
  $Y \to aaYbHKv$
  sono non terminali e hanno lo stesso pattern

Pattern

Data una produzione $X \to v_0 X_1 v_1 \dots v_{n-1} X_n v_n$ dove $v_0, \dots, v_n$ sono stringhe (anche vuote) di terminali
il suo **pattern** è la stringa $v_0 \# v_1 \# v_2 \# \dots \# v_{n-1} \# v_n$
... ossia è la stringa dei terminali tra le variabili

- Le produzioni
  $X \to aaXbYZv$
  $Y \to aaYbHKv$
  sono non terminali e hanno lo stesso pattern
- Le seguenti produzioni sono terminali e hanno stesso pattern
  $X \to bbcd$
  $Y \to bbcd$

Esempio di normalizzazione

Consideriamo $L_3 = \{0^n 1^m 2^{n+m} : n + m > 0\}$ che è generato dalla seguente grammatica che non rispetta la forma normale del lemma:
$S \to 02 \mid 0S2 \mid B$
$B \to 12 \mid 1B2$

Esempio di normalizzazione

Consideriamo $L_3 = \{0^n 1^m 2^{n+m} : n + m > 0\}$ che è generato dalla seguente grammatica che non rispetta la forma normale del lemma:
$S \to 02 \mid 0S2 \mid B$
$B \to 12 \mid 1B2$
possiamo costruire una nuova grammatica normalizzata che genera lo stesso linguaggio:

Esempio di normalizzazione

Consideriamo $L_3 = \{0^n 1^m 2^{n+m} : n + m > 0\}$ che è generato dalla seguente grammatica che non rispetta la forma normale del lemma:
$S \rightarrow 02 \mid 0S2 \mid B$
$B \rightarrow 12 \mid 1B2$
possiamo costruire una nuova grammatica normalizzata che genera lo stesso linguaggio:
$S \rightarrow 02 \mid 12 \mid 1122$
$S \rightarrow 0S2 \mid 11B22$
$B \rightarrow 12$
$B \rightarrow 1B2$

Logica per linguaggi Context-Free

Definire una logica per i linguaggi Context-Free

- Cerchiamo qualcosa di più potente rispetto a *MSO*
- $\exists SO$ è troppo potente:

Definire una logica per i linguaggi Context-Free

- Cerchiamo qualcosa di più potente rispetto a *MSO*
- $\exists SO$ è troppo potente:
  - Dal teorema di Fagin (1974) sappiamo che cattura *NP*
  - $L_4 = \{a^n b^n c^n : n > 0\}$ è decidibile in tempo lineare (quindi banalmente $L_4 \in NP$) ma non è CF

Definire una logica per i linguaggi Context-Free

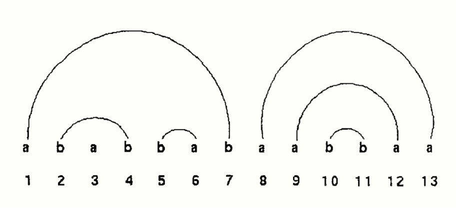
- Cerchiamo qualcosa di più potente rispetto a *MSO*
- $\exists SO$ è troppo potente:
  - Dal teorema di Fagin (1974) sappiamo che cattura *NP*
  - $L_4 = \{a^n b^n c^n : n > 0\}$ è decidibile in tempo lineare (quindi banalmente $L_4 \in NP$) ma non è CF
- Anche una logica del tipo $\exists R\ FO$ dove R è binaria è troppo potente ...anche in questo caso riusciamo a caratterizzare L_4
- Cerchiamo qualcosa nel mezzo...

Definire una logica per i linguaggi Context-Free

- Cerchiamo qualcosa di più potente rispetto a MSO
- $\exists SO$ è troppo potente:
  - Dal teorema di Fagin (1974) sappiamo che cattura NP
  - $L_4 = \{a^n b^n c^n : n > 0\}$ è decidibile in tempo lineare (quindi banalmente $L_4 \in NP$) ma non è CF
- Anche una logica del tipo $\exists R\ FO$ dove R è binaria è troppo potente ...anche in questo caso riusciamo a caratterizzare L_4
- Cerchiamo qualcosa nel mezzo...

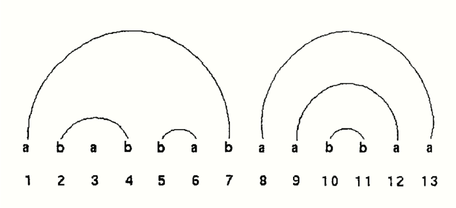
Mostreremo che i linguaggi CF (che non contengono la parola vuota) sono catturati da una logica del tipo $\exists M\ FO$ dove M è un **matching**

Matching



Un **matching M** è una particolare relazione binaria.

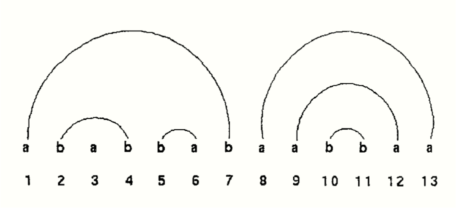
Matching



Un **matching M** è una particolare relazione binaria.

1. $\forall i \forall j (M(i,j) \implies i < j)$.

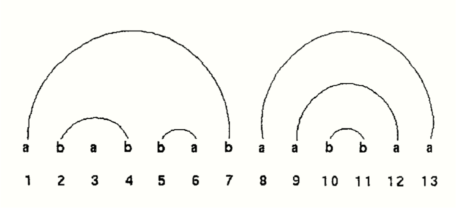
Matching



Un **matching M** è una particolare relazione binaria.

1. $\forall i \forall j (M(i,j) \implies i < j)$.
2. Ogni elemento appartiene ad al massimo una coppia ossia $\forall i \forall j \forall k ((M(i,j) \wedge (k \neq i \wedge k \neq j)) \implies (\neg M(i,k) \wedge \neg M(k,i) \wedge \neg M(j,k) \wedge \neg M(k,j)))$

Matching



Un **matching M** è una particolare relazione binaria.

1. $\forall i \forall j (M(i,j) \implies i < j)$.
2. Ogni elemento appartiene ad al massimo una coppia ossia $\forall i \forall j \forall k ((M(i,j) \land (k \neq i \land k \neq j)) \implies (\neg M(i,k) \land \neg M(k,i) \land \neg M(j,k) \land \neg M(k,j)))$
3. Non ci sono *incroci* ossia $\forall i \forall j \forall k \forall l (M(i,j) \land M(k,l) \land i < k < j \implies i < l < j)$

Caratterizzazione linguaggio CF: esempio 1

$L_1 = \{a^n b^n : n > 0\}$
Definiamo F_{L1} tale che:

Caratterizzazione linguaggio CF: esempio 1

$L_1 = \{a^n b^n : n > 0\}$

Definiamo F_{L1} tale che:

- Ogni elemento è catturato dal matching

Caratterizzazione linguaggio CF: esempio 1

$L_1 = \{a^n b^n : n > 0\}$

Definiamo F_{L1} tale che:

- Ogni elemento è catturato dal matching
- Ogni a è matchata con una b

Caratterizzazione linguaggio CF: esempio 1

$L_1 = \{a^n b^n : n > 0\}$

Definiamo F_{L1} tale che:

- Ogni elemento è catturato dal matching
- Ogni a è matchata con una b
- Prima troviamo tutte le a e poi tutte le b

Caratterizzazione linguaggio CF: esempio 1

$L_1 = \{a^n b^n : n > 0\}$
Definiamo F_{L1} tale che:

- Ogni elemento è catturato dal matching
- Ogni a è matchata con una b
- Prima troviamo tutte le a e poi tutte le b

$$F_{L1} := \exists M[$$

Caratterizzazione linguaggio CF: esempio 1

$L_1 = \{a^n b^n : n > 0\}$
Definiamo F_{L1} tale che:

- Ogni elemento è catturato dal matching
- Ogni a è matchata con una b
- Prima troviamo tutte le a e poi tutte le b

$$F_{L1} := \exists M[$$
$$\forall x \exists z (M(x,z) \vee M(z,x)) \wedge$$

Caratterizzazione linguaggio CF: esempio 1

$L_1 = \{a^n b^n : n > 0\}$
Definiamo F_{L1} tale che:

- Ogni elemento è catturato dal matching
- Ogni a è matchata con una b
- Prima troviamo tutte le a e poi tutte le b

$$
\begin{aligned}
F_{L1} := \exists M[\\
&\forall x \exists z (M(x,z) \vee M(z,x)) \wedge \\
&\forall x \forall y (M(x,y) \implies a(x) \wedge b(y)) \wedge
\end{aligned}
$$

Caratterizzazione linguaggio CF: esempio 1

$L_1 = \{a^n b^n : n > 0\}$
Definiamo F_{L1} tale che:

- Ogni elemento è catturato dal matching
- Ogni a è matchata con una b
- Prima troviamo tutte le a e poi tutte le b

$$
\begin{aligned}
F_{L1} := \exists M[\\
&\forall x \exists z (M(x,z) \vee M(z,x)) \wedge \\
&\forall x \forall y (M(x,y) \implies a(x) \wedge b(y)) \wedge \\
&\forall x \forall y \forall u \forall v (M(x,y) \wedge M(u,v) \implies u < y) \\
&]
\end{aligned}
$$

Caratterizzazione linguaggio CF: esempio 1

$L_1 = \{a^n b^n : n > 0\}$
Definiamo F_{L1} tale che:

- Ogni elemento è catturato dal matching
- Ogni a è matchata con una b
- Prima troviamo tutte le a e poi tutte le b

$$
\begin{aligned}
F_{L1} := \exists M[& \\
& \forall x \exists z (M(x,z) \vee M(z,x)) \wedge \\
& \forall x \forall y (M(x,y) \implies a(x) \wedge b(y)) \wedge \\
& \forall x \forall y \forall u \forall v (M(x,y) \wedge M(u,v) \implies u < y) \\
&]
\end{aligned}
$$

abbiamo
$w \in L_1 \iff w \models F_{L1}$

Caratterizzazione linguaggio CF: esempio 2

$L_3 = \{0^n 1^m 2^{n+m} : n + m > 0\}$
possiamo definire la formula F_{L3} come segue

Caratterizzazione linguaggio CF: esempio 2

$L_3 = \{0^n 1^m 2^{n+m} : n + m > 0\}$
possiamo definire la formula F_{L3} come segue

$F_{L3} := \exists M[$

Caratterizzazione linguaggio CF: esempio 2

$L_3 = \{0^n 1^m 2^{n+m} : n + m > 0\}$
possiamo definire la formula F_{L3} come segue

$$F_{L3} := \exists M[$$
$$\forall x \exists z (M(x,z) \vee M(z,x)) \wedge$$

Caratterizzazione linguaggio CF: esempio 2

$L_3 = \{0^n 1^m 2^{n+m} : n + m > 0\}$
possiamo definire la formula F_{L3} come segue

$$
\begin{aligned}
&F_{L3} := \exists M[\\
&\quad \forall x \exists z (M(x, z) \vee M(z, x)) \wedge \\
&\quad \exists p \forall x \forall y (M(x, y) \implies 2(y) \wedge ((x \leq p \wedge 0(x)) \vee (x > p \wedge 1(x)))) \wedge
\end{aligned}
$$

Caratterizzazione linguaggio CF: esempio 2

$L_3 = \{0^n 1^m 2^{n+m} : n + m > 0\}$
possiamo definire la formula F_{L3} come segue

$$
\begin{aligned}
& F_{L3} := \exists M[\\
& \quad \forall x \exists z (M(x,z) \vee M(z,x)) \wedge \\
& \quad \exists p \forall x \forall y (M(x,y) \implies 2(y) \wedge ((x \leq p \wedge 0(x)) \vee (x > p \wedge 1(x)))) \wedge \\
& \quad \forall x \forall y \forall u \forall v (M(x,y) \wedge M(u,v) \implies u < y) \\
& \quad]
\end{aligned}
$$

Caratterizzazione linguaggio CF: esempio 2

$L_3 = \{0^n 1^m 2^{n+m} : n + m > 0\}$
possiamo definire la formula F_{L3} come segue

$$
\begin{aligned}
&F_{L3} := \exists M[\\
&\quad \forall x \exists z (M(x,z) \vee M(z,x)) \wedge\\
&\quad \exists p \forall x \forall y (M(x,y) \implies 2(y) \wedge ((x \leq p \wedge 0(x)) \vee (x > p \wedge 1(x)))) \wedge\\
&\quad \forall x \forall y \forall u \forall v (M(x,y) \wedge M(u,v) \implies u < y)\\
&\quad]
\end{aligned}
$$

abbiamo
$w \in L_3 \iff w \models F_{L3}$

Teorema principale (prima parte)

Teorema

$CF(\Sigma) \subseteq \exists M \; FO(=, <, M, (a(x))_{a \in \Sigma})$ *dove M è un matching*

Teorema principale (prima parte)

Teorema

$CF(\Sigma) \subseteq \exists M \; FO(=, <, M, (a(x))_{a \in \Sigma})$ *dove* M *è un matching*

Ossia vogliamo dire che per ogni grammatica Context-Free G (messa nella forma normale vista in precedenza) su un alfabeto Σ, esiste una formula $\phi_G = \exists M \psi_G$ dove $\psi_G \in FO(=, <, M, (a(x))_{a \in \Sigma})$ tale che:

$$w \models \phi_G \iff w \text{ può essere generata da } G$$

Proof $CF \subseteq \exists M\ FO$ (1)

Siano

- L un linguaggio
- G una grammatica CF (nella forma normale del lemma precedente) che caratterizza L
- $w \in L$ una parola

Denotiamo con $G(w)$ un albero di derivazione di w a paritre da G

Proof $CF \subseteq \exists M\ FO$ (1)

Siano

- L un linguaggio
- G una grammatica CF (nella forma normale del lemma precedente) che caratterizza L
- $w \in L$ una parola

Denotiamo con $G(w)$ un albero di derivazione di w a paritre da G

Schema della dimostrazione

1. Ad ogni albero $G(w)$ corrisponde un matching $M_{G(w)}$
2. Dato il matching $M_{G(w)}$ e la parola w siamo in grado di riconoscere le produzioni e ricostruire l'albero

Proof $CF \subseteq \exists M\ FO$ (1)

Siano

- L un linguaggio
- G una grammatica CF (nella forma normale del lemma precedente) che caratterizza L
- $w \in L$ una parola

Denotiamo con $G(w)$ un albero di derivazione di w a paritre da G

Schema della dimostrazione

1. Ad ogni albero $G(w)$ corrisponde un matching $M_{G(w)}$
2. Dato il matching $M_{G(w)}$ e la parola w siamo in grado di riconoscere le produzioni e ricostruire l'albero
3. Esiste una formula $\psi_G(M)$ tale che

$$\begin{array}{c} w \models \psi_G(M) \\ \text{con } M \text{ matching} \end{array} \iff \begin{array}{l} \text{esiste albero di derivazione} \\ G(w) \text{ tale che } M = M_{G(w)} \end{array}$$

Proof $CF \subseteq \exists M\ FO$ (2)

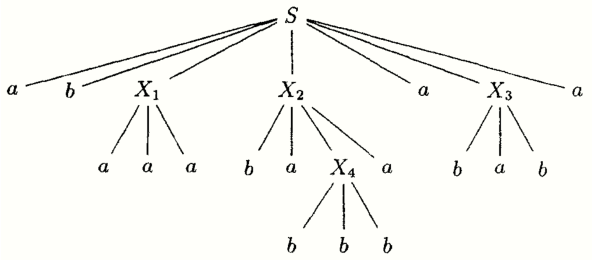
Consideriamo la seguente grammatica G, sia $w = abaaababbbaababa$, mostriamo $G(w)$

$S \rightarrow abX_1X_2aX_3a \mid \dots$
$X_1 \rightarrow aaa \mid \dots$
$X_2 \rightarrow baX_4a \mid \dots$
$X_3 \rightarrow bab \mid \dots$
$X_4 \rightarrow bbb \mid \dots$

Proof $CF \subseteq \exists M\ FO$ (2)

Consideriamo la seguente grammatica G, sia $w = abaaababbbaababa$, mostriamo $G(w)$

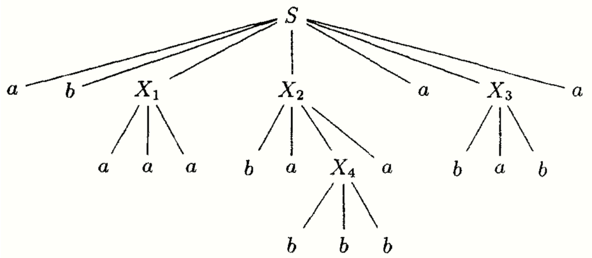
$S \to abX_1X_2aX_3a \mid \dots$
$X_1 \to aaa \mid \dots$
$X_2 \to baX_4a \mid \dots$
$X_3 \to bab \mid \dots$
$X_4 \to bbb \mid \dots$



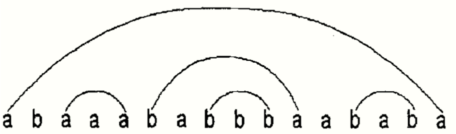
Proof $CF \subseteq \exists M\ FO$ (2)

Consideriamo la seguente grammatica G, sia $w = abaaababbbaababa$, mostriamo $G(w)$

$$
\begin{aligned}
S &\rightarrow abX_1X_2aX_3a \mid \ldots \\
X_1 &\rightarrow aaa \mid \ldots \\
X_2 &\rightarrow baX_4a \mid \ldots \\
X_3 &\rightarrow bab \mid \ldots \\
X_4 &\rightarrow bbb \mid \ldots
\end{aligned}
$$



Costruiamo il matching $M_{G(w)}$ mettendo in relazione il figlio più a destra e il figlio più a sinistra di ogni nodo interno

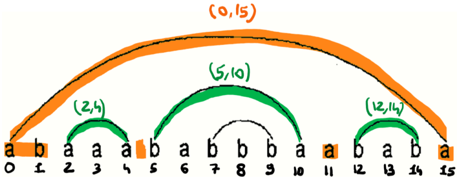


Proof $CF \subseteq \exists M\ FO$ (3)

Dati w e il matching $M_{G(w)}$ possiamo identificare i pattern delle produzioni da cui w viene generata

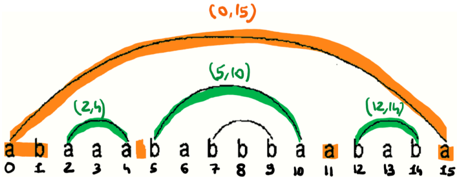
Proof $CF \subseteq \exists M\ FO$ (3)

Dati w e il matching $M_{G(w)}$ possiamo identificare i pattern delle produzioni da cui w viene generata



Proof $CF \subseteq \exists M\ FO$ (3)

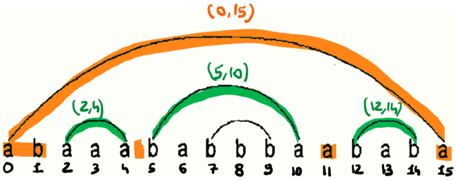
Dati w e il matching $M_{G(w)}$ possiamo identificare i pattern delle produzioni da cui w viene generata



- ab # ε # a # a è il pattern dell'arco $(0, 15)$ nella stringa w

Proof $CF \subseteq \exists M\ FO$ (3)

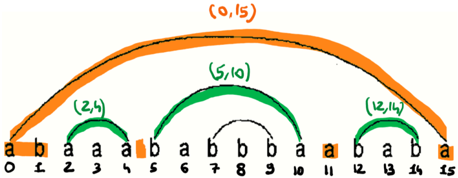
Dati w e il matching $M_{G(w)}$ possiamo identificare i pattern delle produzioni da cui w viene generata



- ab # ε # a # a è il pattern dell'arco $(0, 15)$ nella stringa w
- $ab\#\varepsilon\#a\#a$ è il pattern della produzione $S \rightarrow abX_1X_2aX_3a$

Proof $CF \subseteq \exists M\ FO$ (3)

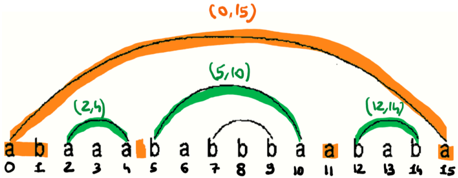
Dati w e il matching $M_{G(w)}$ possiamo identificare i pattern delle produzioni da cui w viene generata



- ab # ε # a # a è il pattern dell'arco (0, 15) nella stringa w
- $ab\#\varepsilon\#a\#a$ è il pattern della produzione $S \rightarrow abX_1X_2aX_3a$
- L'arco (0, 15) **corrisponde** alla produzione $S \rightarrow abX_1X_2aX_3a$

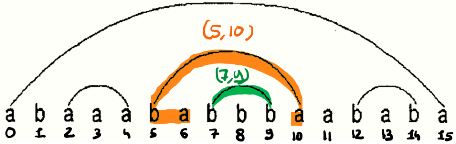
Proof $CF \subseteq \exists M\ FO$ (3)

Dati w e il matching $M_{G(w)}$ possiamo identificare i pattern delle produzioni da cui w viene generata

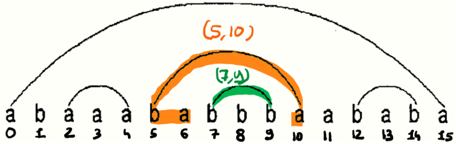


- ab # ε # a # a è il pattern dell'arco $(0, 15)$ nella stringa w
- $ab\#\varepsilon\#a\#a$ è il pattern della produzione $S \rightarrow abX_1X_2aX_3a$
- L'arco $(0, 15)$ **corrisponde** alla produzione $S \rightarrow abX_1X_2aX_3a$
- Gli archi $(2, 4)$, $(5, 10)$, $(12, 14)$ sono nella **superficie** dell'arco $(0, 15)$

Proof $CF \subseteq \exists M\ FO$ (4)



Proof $CF \subseteq \exists M\ FO$ (4)



- L'arco $(5, 10)$ (che ha pattern ba $\#$ a) **corrisponde** alla produzione $X_2 \rightarrow baX_4a$
- L'arco $(7, 9)$ è nella **superficie** dell'arco $(5, 10)$

Proof $CF \subseteq \exists M\ FO$ (5)

- Alle produzioni corrispondono dei pattern

Proof $CF \subseteq \exists M\ FO$ (5)

- Alle produzioni corrispondono dei pattern
- Data una grammatica G e una parola w, possiamo costruire il matching $M_{G(w)}$ relativo all'albero di derivazione $G(w)$

Proof $CF \subseteq \exists M\ FO$ (5)

- Alle produzioni corrispondono dei pattern
- Data una grammatica G e una parola w, possiamo costruire il matching $M_{G(w)}$ relativo all'albero di derivazione $G(w)$
- Data una parola w e un matching M abbiamo visto i pattern indotti dagli archi del matching

Proof $CF \subseteq \exists M\ FO$ (5)

- Alle produzioni corrispondono dei pattern
- Data una grammatica G e una parola w, possiamo costruire il matching $M_{G(w)}$ relativo all'albero di derivazione $G(w)$
- Data una parola w e un matching M abbiamo visto i pattern indotti dagli archi del matching
- Troviamo ora la formula $\psi_G(M)$ tale che

$$\begin{array}{c} w \models \psi_G(M) \\ \text{con } M \text{ matching} \end{array} \iff \begin{array}{l} \text{esiste albero di derivazione} \\ G(w) \text{ tale che } M = M_{G(w)} \end{array}$$

Proof $CF \subseteq \exists M\ FO$ (5)

- Alle produzioni corrispondono dei pattern
- Data una grammatica G e una parola w, possiamo costruire il matching $M_{G(w)}$ relativo all'albero di derivazione $G(w)$
- Data una parola w e un matching M abbiamo visto i pattern indotti dagli archi del matching
- Troviamo ora la formula $\psi_G(M)$ tale che

$$\begin{array}{c} w \models \psi_G(M) \\ \text{con } M \text{ matching} \end{array} \iff \begin{array}{l} \text{esiste albero di derivazione} \\ G(w) \text{ tale che } M = M_{G(w)} \end{array}$$

- La formula metterà in corrispondenza i pattern degli archi del matching M e i pattern delle produzioni di G
- Se la corrispondenza esisterà allora $w \in L$

Proof $CF \subseteq \exists M\ FO$ (6)

- $S_v(x, y) :=$ " tra le posizioni x e y (escluse) si trova la stringa v "

Proof $CF \subseteq \exists M\ FO$ (6)

- $S_v(x, y) :=$ " tra le posizioni x e y (escluse) si trova la stringa v "
- H_u è soddisfatta solo dalle stringhe che sono esattamente u

Proof $CF \subseteq \exists M\ FO$ (6)

- $S_v(x, y) :=$ " tra le posizioni x e y (escluse) si trova la stringa v "
- H_u è soddisfatta solo dalle stringhe che sono esattamente u
- $F_p(x, y) :=$ " l'arco (x, y) corrisponde alla produzione p "
  dove p è la produzione $X_0 \to \alpha v_0 X_1 v_1 X_2 \dots v_{s-1} X_s v_s \beta$

Proof $CF \subseteq \exists M\ FO$ (6)

- $S_v(x, y) :=$ " tra le posizioni x e y (escluse) si trova la stringa v "
- H_u è soddisfatta solo dalle stringhe che sono esattamente u
- $F_p(x, y) :=$ " l'arco (x, y) corrisponde alla produzione p "
  dove p è la produzione $X_0 \to \alpha v_0 X_1 v_1 X_2 \dots v_{s-1} X_s v_s \beta$

$$F_p(x, y) := \alpha(x) \wedge \beta(y) \wedge \exists x_1 \exists y_1 \dots \exists x_s \exists y_s [$$
$$(x < x_1 < y_1 < x_2 \dots < y_s < y) \wedge$$

Proof $CF \subseteq \exists M\ FO$ (6)

- $S_v(x, y) :=$ " tra le posizioni x e y (escluse) si trova la stringa v "
- H_u è soddisfatta solo dalle stringhe che sono esattamente u
- $F_p(x, y) :=$ " l'arco (x, y) corrisponde alla produzione p "
  dove p è la produzione $X_0 \to \alpha v_0 X_1 v_1 X_2 \dots v_{s-1} X_s v_s \beta$

$$\begin{aligned} F_p(x, y) := \alpha(x) \wedge \beta(y) \wedge \exists x_1 \exists y_1 \dots \exists x_s \exists y_s [\\ (x < x_1 < y_1 < x_2 \dots < y_s < y) \wedge \\ (S_{v_0}(x, x_1) \wedge S_{v_1}(y_1, x_2) \wedge \dots \wedge S_{v_s}(y_s, y)) \wedge \end{aligned}$$

Proof $CF \subseteq \exists M\ FO$ (6)

- $S_v(x, y) :=$ " tra le posizioni x e y (escluse) si trova la stringa v "
- H_u è soddisfatta solo dalle stringhe che sono esattamente u
- $F_p(x, y) :=$ " l'arco (x, y) corrisponde alla produzione p "
  dove p è la produzione $X_0 \to \alpha v_0 X_1 v_1 X_2 \dots v_{s-1} X_s v_s \beta$

$$
\begin{aligned}
F_p(x, y) := \alpha(x) \wedge \beta(y) \wedge \exists x_1 \exists y_1 \dots \exists x_s \exists y_s [\\
& (x < x_1 < y_1 < x_2 \dots < y_s < y) \wedge \\
& (S_{v_0}(x, x_1) \wedge S_{v_1}(y_1, x_2) \wedge \dots \wedge S_{v_s}(y_s, y)) \wedge \\
& (M(x_1, y_1) \wedge \dots M(x_s, y_s))]
\end{aligned}
$$

Proof $CF \subseteq \exists M\ FO$ (6)

- $S_v(x, y) :=$ " tra le posizioni x e y (escluse) si trova la stringa v "
- H_u è soddisfatta solo dalle stringhe che sono esattamente u
- $F_p(x, y) :=$ " l'arco (x, y) corrisponde alla produzione p "
  dove p è la produzione $X_0 \rightarrow \alpha v_0 X_1 v_1 X_2 \dots v_{s-1} X_s v_s \beta$

$$
\begin{aligned}
F_p(x, y) := \alpha(x) \wedge \beta(y) \wedge \exists x_1 \exists y_1 \dots \exists x_s \exists y_s [\\
(x < x_1 < y_1 < x_2 \dots < y_s < y) \wedge \\
(S_{v_0}(x, x_1) \wedge S_{v_1}(y_1, x_2) \wedge \dots \wedge S_{v_s}(y_s, y)) \wedge \\
(M(x_1, y_1) \wedge \dots M(x_s, y_s))]
\end{aligned}
$$

- $\tilde{F}_X(x, y) :=$ " l'arco (x, y) corrisponde ad una produzione da X "
  è composta dalla disgiunzione di formule $F_p(x, y)$

Proof $CF \subseteq \exists M\ FO$ (6)

- $S_v(x, y) :=$ " tra le posizioni x e y (escluse) si trova la stringa v "
- H_u è soddisfatta solo dalle stringhe che sono esattamente u
- $F_p(x, y) :=$ " l'arco (x, y) corrisponde alla produzione p "
  dove p è la produzione $X_0 \rightarrow \alpha v_0 X_1 v_1 X_2 \dots v_{s-1} X_s v_s \beta$

$$\begin{aligned} F_p(x, y) := \alpha(x) \wedge \beta(y) \wedge \exists x_1 \exists y_1 \dots \exists x_s \exists y_s [\\ (x < x_1 < y_1 < x_2 \cdots < y_s < y) \wedge \\ (S_{v_0}(x, x_1) \wedge S_{v_1}(y_1, x_2) \wedge \cdots \wedge S_{v_s}(y_s, y)) \wedge \\ (M(x_1, y_1) \wedge \dots M(x_s, y_s))] \end{aligned}$$

- $\tilde{F}_X(x, y) :=$ " l'arco (x, y) corrisponde ad una produzione da X "
  è composta dalla disgiunzione di formule $F_p(x, y)$
- $\hat{F}_p(x, y) :=$ " l'arco (x, y) corrisponde alla produzione p **e** gli archi di superficie di (x, y) corrispondono a produzioni da $X_1, \dots, X_s$ "

Proof $CF \subseteq \exists M\ FO$ (7)

$$\psi_G(M) := \bigvee_{S \to u \in P} H_u \vee [(M(min, max) \wedge \tilde{F}_S(min, max)) \wedge$$

Proof $CF \subseteq \exists M\ FO$ (7)

$$\psi_G(M) := \bigvee_{S \to u \in P} H_u \vee [(M(min, max) \wedge \tilde{F}_S(min, max)) \wedge$$

$$(\forall x \forall y (M(x, y) \implies \bigvee_{p \in P} \hat{F}_p(x, y)))]$$

Proof $CF \subseteq \exists M\ FO$ (7)

$$\psi_G(M) := \bigvee_{S \to u \in P} H_u \vee [(M(min, max) \wedge \tilde{F}_S(min, max)) \wedge$$
$$(\forall x \forall y (M(x, y) \implies \bigvee_{p \in P} \hat{F}_p(x, y)))]$$

- Se w può essere derivata in G, il matching $M_{G(w)}$ è tale che $w \models \psi_G(M_{G(w)})$

Proof $CF \subseteq \exists M\ FO$ (7)

$$\psi_G(M) := \bigvee_{S \to u \in P} H_u \vee [(M(min, max) \wedge \tilde{F}_S(min, max)) \wedge (\forall x \forall y (M(x,y) \implies \bigvee_{p \in P} \hat{F}_p(x,y)))]$$

- Se w può essere derivata in G, il matching $M_{G(w)}$ è tale che $w \models \psi_G(M_{G(w)})$
- Viceversa, si mostra induttivamente che se $w \models \psi_G(M)$ per un qualche M e vale $M(i,j)$ allora $w_i \dots w_j$ può essere derivata da un qualche simbolo non terminale X.
  Siccome vale $M(min, max)$ e (min, max) corrisponde ad una produzione da S, allora w può essere derivata in G.

$\square$

Esempio applicazione del teorema

$L_3 = \{0^n 1^m 2^{n+m} : n + m > 0\}$ è generato da

$$\overset{s_1}{S \to 02} \quad \overset{s_2}{S \to 12} \quad \overset{s_3}{S \to 1122} \quad \overset{s_4}{S \to 0S2} \quad \overset{s_5}{S \to 11B22}$$

$$\underset{b_1}{B \to 12} \quad \underset{b_2}{B \to 1B2}$$

Esempio applicazione del teorema

$L_3 = \{0^n 1^m 2^{n+m} : n + m > 0\}$ è generato da

$$\overset{s_1}{S \to 02} \quad \overset{s_2}{S \to 12} \quad \overset{s_3}{S \to 1122} \quad \overset{s_4}{S \to 0S2} \quad \overset{s_5}{S \to 11B22}$$

$$\underset{b_1}{B \to 12} \quad \underset{b_2}{B \to 1B2}$$

$$\psi_{L3}(M) := H_{02} \vee H_{12} \vee H_{1122} \vee [(\forall x \forall y (M(x,y) \implies \hat{F}_{s_1}(x,y) \vee \cdots \vee \hat{F}_{s_5}(x,y) \vee \hat{F}_{b_1}(x,y) \vee \hat{F}_{b_2}(x,y))) \wedge (M(min, max) \wedge \tilde{F}_S(min, max))]$$

Esempio applicazione del teorema

$L_3 = \{0^n 1^m 2^{n+m} : n + m > 0\}$ è generato da

$$\overset{s_1}{S \to 02} \quad \overset{s_2}{S \to 12} \quad \overset{s_3}{S \to 1122} \quad \overset{s_4}{S \to 0S2} \quad \overset{s_5}{S \to 11B22}$$

$$\underset{b_1}{B \to 12} \quad \underset{b_2}{B \to 1B2}$$

$$\psi_{L3}(M) := H_{02} \vee H_{12} \vee H_{1122} \vee [(\forall x \forall y (M(x,y) \implies \hat{F}_{s_1}(x,y) \vee \cdots \vee \hat{F}_{s_5}(x,y) \vee \hat{F}_{b_1}(x,y) \vee \hat{F}_{b_2}(x,y))) \wedge (M(min, max) \wedge \tilde{F}_S(min, max))]$$

$$\hat{F}_{s_5}(x,y) := 1(x) \wedge 2(y) \wedge \exists x_1 \exists y_1 [(x < x_1 < y_1 < y) \wedge S_1(x, x_1) \wedge S_2(y_1, y) \wedge M(x_1, y_1) \wedge \tilde{F}_B(x_1, y_1)]$$

Teorema principale (seconda parte)

Teorema

$\exists M \ FO(=, <, M, (a(x))_{a \in \Sigma}) \subseteq CF(\Sigma)$ *dove M è un matching*

Conclusioni

Remark conclusivi

- I linguaggi CF si collocano appena sopra ai regolari
- Dal punto di vista degli automi, i linguaggi CF sono riconosciuti da degli automi con una memoria, la pila
- Dal punto di vista logico i linguaggi CF sono catturati dalla nozione di matching
- Per dimostrare che $CF(\Sigma) \subseteq \exists M\ FO(=, <, M, (a(x))_{a\in\Sigma})$ abbiamo visto come far corrispondere i pattern indotti dal matching ai pattern delle produzioni

Bibliografia

Dovier, A. (2020).
Fondamenti dell'informatica : linguaggi formali, calcolabilità e complessità / Agostino Dovier, Roberto Giacobazzi.
Programma di matematica, fisica, elettronica. Bollati Boringhieri, Torino.

Lautemann, C., Schwentick, T., and Thérien, D. (1995).
Logics for context-free languages.
In Pacholski, L. and Tiuryn, J., editors, *Computer Science Logic*, pages 205–216, Berlin, Heidelberg. Springer Berlin Heidelberg.

Domande?