Compilatori

Compilatori

TODO Fino a Secondo Pdf a Pag 29

14/10/2025 Ambiguità Inerente Fino a Pag 51

Un CFL (Context-Free Languages) è inerentemente ambiguo se tutte le grammatiche per L sono ambigue.
Esempio:

$$\{a^n b^n c^m d^m : n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^m : n \geq 1, m \geq 1\}$$

Una grammatica per L e'

$$\begin{aligned}
S &\rightarrow AB \mid C \\
A &\rightarrow aAb \mid ab \\
B &\rightarrow cBd \mid cd \\
C &\rightarrow aCd \mid aDd \\
D &\rightarrow bDc \mid bc
\end{aligned}$$

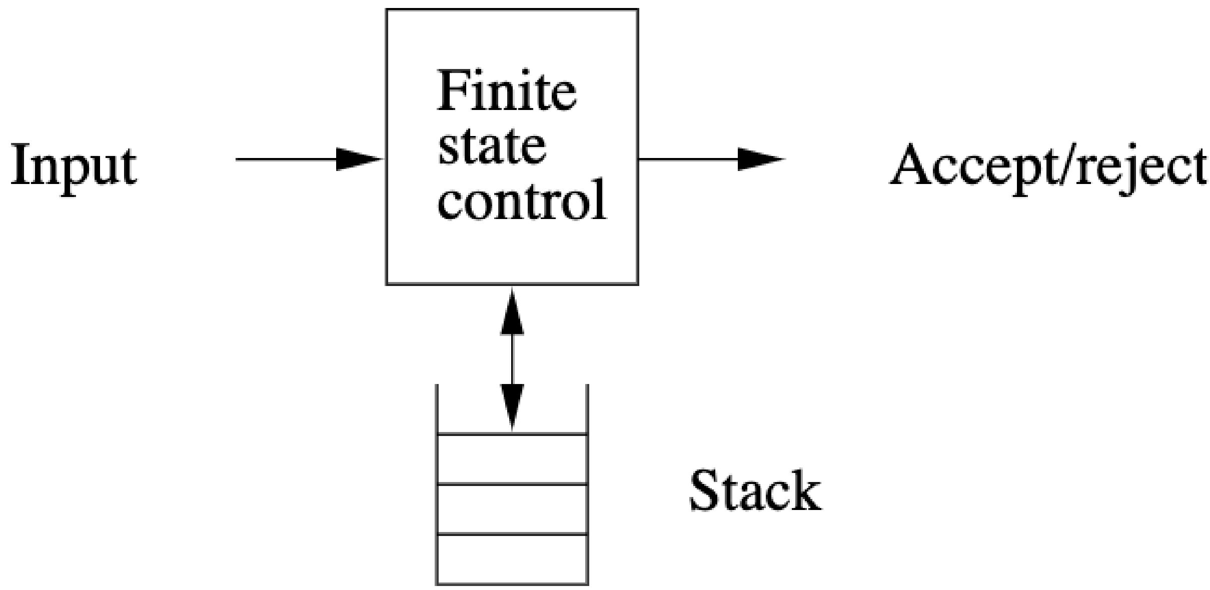
Automi a Pila (Push Down Automaton)

> Da immaginare lo Stack sdraiato da sinistra a destra(sinistra next pop).

Un automa a pila PDA è in pratica un automa a stati finiti (ϵ-NFA che è la versione più estesa deli automi a stati finiti) con una pila (struttura dati).
Non devono essere deterministici.
IMPORTANTE => è una pila perchè l'ultimo blocco che incontriamo è quello da cui dobbiamo partire, perchè funzioniona come un xml o comunque

come una gestione a tag (html) quindi ha senso che sia una stack anzichè una queue.

1. Consuma un simbolo di input o esegue una transizione ϵ.
2. Va in un nuovo stato (o rimane dove e').
3. Rimpiazza il top della pila con una stringa (consuma il carattere in cima, e mette al suo posto una stringa, eventualmente vuota o uguale al carattere consumato lasciando quindi la pila inalterata) $\rightarrow$ fa un pop ed una push per rimpiazzare con un carattere scelto da noi!



Esempio:

$$L_{wwr} = \{ww^r : w \in \{0,1\}^*\}$$

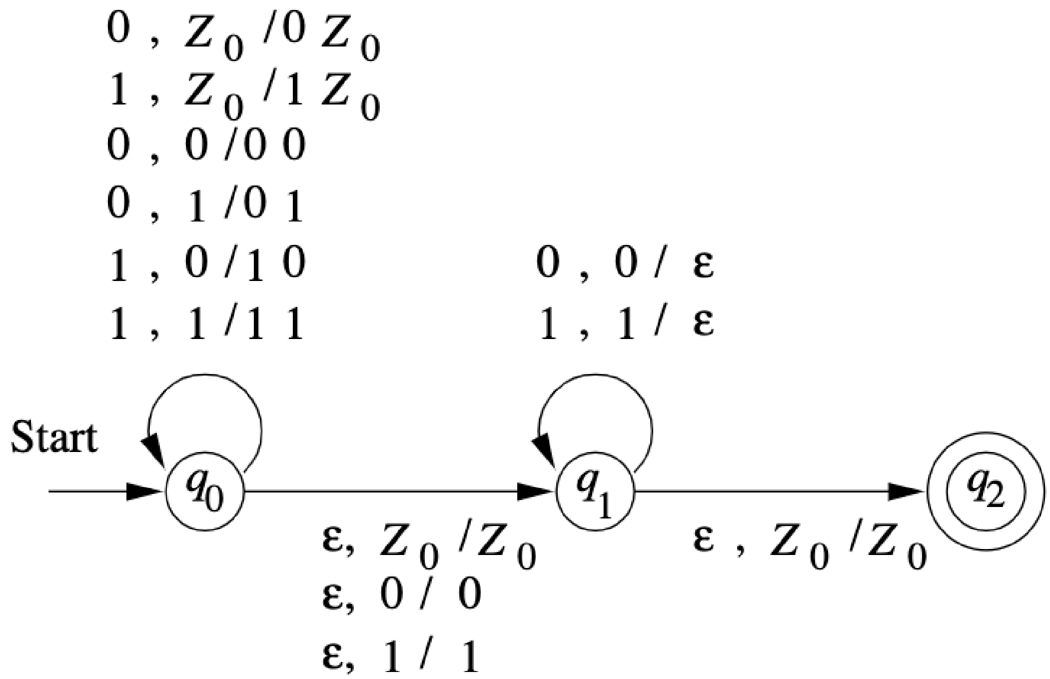
con "grammatica" $P \to 0P0$, $P \to 1P1$, $P \to \epsilon$. Un PDA per L_{wwr} ha **tre stati**, e funziona come segue:

1. Legge w un simbolo alla volta, rimanendo nello stato q_0, e aggiungendo il simbolo di input alla pila.
2. Decide non deterministicamente che sta nel mezzo di ww^R e va nello stato q_1.
3. Legge w^R un simbolo alla volta e lo paragona col simbolo al top della pila: Se sono uguali, fa un pop della pila, e rimane nello stato q_1. Se non sono uguali, si blocca.
4. Se la pila non ha piu' simboli (0 o 1), va nello stato q_2 e accetta.

Punto 2 $\to$ il PDA ad ogni carattere prende due strade, la prima per provare a capire se è in mezzo non deterministica va nello stato q_1 e prova a matchare ww^r con il primo elemento dello stack, se c'è un mismatch si blocca.

> UNA STRINGA è accettata quando sono in uno stato di accettazione e l'input è finito (è stato tutto "mangiato" dal PDA)

Il PDA per L_{wwr} come diagramma di transizione:



in cima alla pila, dove c'è scritto $0, Z_0/0\ Z_0$ che rappresenta questo:

- il primo zero limita l'input => ci deve essere lo zero in input e ci deve essere lo Z_0 sulla cima della pila
- mentre lo $/0\ Z_0$ rapprenta la stringa da sostituire rendendo la pila in questo modo qui:
  - Posizione 0 $\rightarrow Z_0$
  - Posizione 1 (prossimo pop) $\rightarrow$ 0
  Queste transizioni non vanno a cambiare il contenuto ma anzi lo mantiene dallo stato q_0 allo stato q_1, in maniera tale da andare sempre avanti qualunque sia l'input, questa politica rimane fino al tratto da q_1 a q_1 nel quale avviene ad esempio:
  se c'è zero nell'input, pop del valore 0 che deve essere in cima alla pila e continuo fino ad avere solo Z_0 che mi fa il tratto da q_1 a q_2, quindi in stato di accettazione, ma la stringa potrebbe non essere accetta se l'input a questo punto non è ancora finito.

Significato Di ϵ

ϵ in base a dove è vuol dire "qualsiasi valore" oppure "nulla":

- se è nell'input ad esempio con ϵ , Z_0 $/Z_0$ mi sta a significare $\rightarrow$ "qualsiasi valore ci sia di input e con Z_0 nella cima della pila, *non mangiare nulla dall'input*".
- mentre 0 , 0 $/\epsilon$ vuol dire "0 in input e 0 in cima alla pila, *mangia l'input e non mettere niente nella pila*"

Definizione Formale Di PDA

Un PDA e' una tupla di 7 elementi:

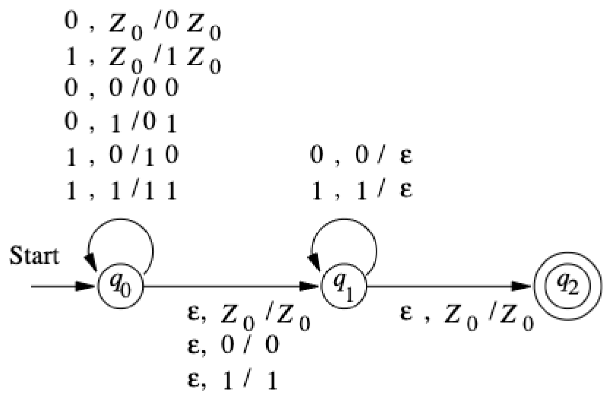
$$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F),$$

dove

- Q e' un insieme finito di stati,
- Σ e' un *alfabeto finito di input,*
- Γ e' un *alfabeto finito di pila,*
- δ e' una *funzione di transizione* da $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma$ a sottinsiemi di $Q \times \Gamma^*$,
- q_0 e' lo *stato iniziale,*
- $Z_0 \in \Gamma$ e' il *simbolo iniziale* per la pila, e
- $F \subseteq Q$ e' l'insieme di *stati di accettazione.*

uguale alla definizione degli ϵ-NFA, in più c'è che la funzione degli ϵ-NFA va da triple a coppie, si parla della funzione δ, in più c'è Γ^* che rappresenta degli sottoinsiemi di stringhe perchè è non deterministico.

Esempio: Il PDA



e' la 7-tupla

$$P = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, Z_0\}, \delta, q_0, Z_0, \{q_2\}),$$

dove δ e' data dalla tabella seguente:

	$0, Z_0$	$1, Z_0$	0,0	0,1	1,0	1,1	ϵ, Z_0	$\epsilon, 0$	$\epsilon, 1$
$\rightarrow q_0$	$\{(q_0, 0Z_0)\}$	$\{(q_0, 1Z_0)\}$	$\{(q_0, 00)\}$	$\{(q_0, 01)\}$	$\{(q_0, 10)\}$	$\{(q_0, 11)\}$	$\{(q_1, Z_0)\}$	$\{(q_1, 0)\}$	$\{(q_1, 1)\}$
q_1	$\emptyset$	$\emptyset$	$\{(q_1, \epsilon)\}$	$\emptyset$	$\emptyset$	$\{(q_1, \epsilon)\}$	$\{(q_2, Z_0)\}$	$\emptyset$	$\emptyset$
$\star q_2$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Descrizioni Istantanee

Un PDA passa da una configurazione ad un'altra configurazione:

- consumando un simbolo di input (o tramite transizione ϵ),
- consumando la cima dello stack sostituendolo con una stringa (eventualmente vuota).

Per ragionare sulle computazioni dei PDA, usiamo delle *descrizioni istantanee* (ID) del PDA. Una ID e' una tripla

$$(q, w, \gamma)$$

dove q e' lo stato, w l'input rimanente, e γ il contenuto della pila.

Sia $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ un PDA. Allora $\forall w \in \Sigma^*, \beta \in \Gamma^*$:

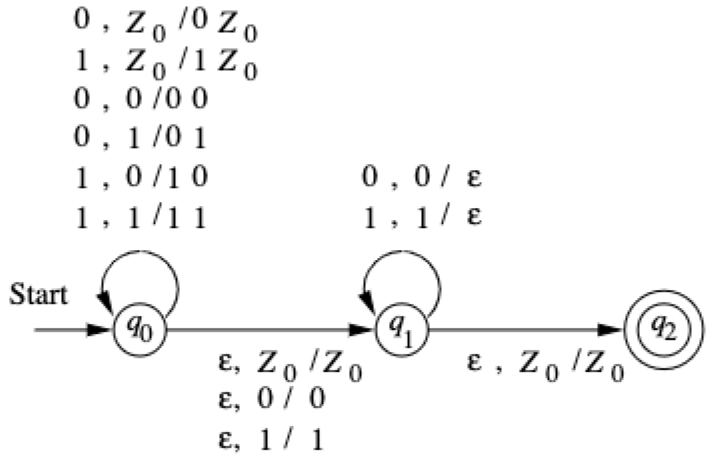
$$(p, \alpha) \in \delta(q, a, X) \Rightarrow (q, aw, X\beta) \vdash (p, w, \alpha\beta).$$

Definiamo $\overset{*}{\vdash}$ la chiusura riflessiva e transitiva di $\vdash$.

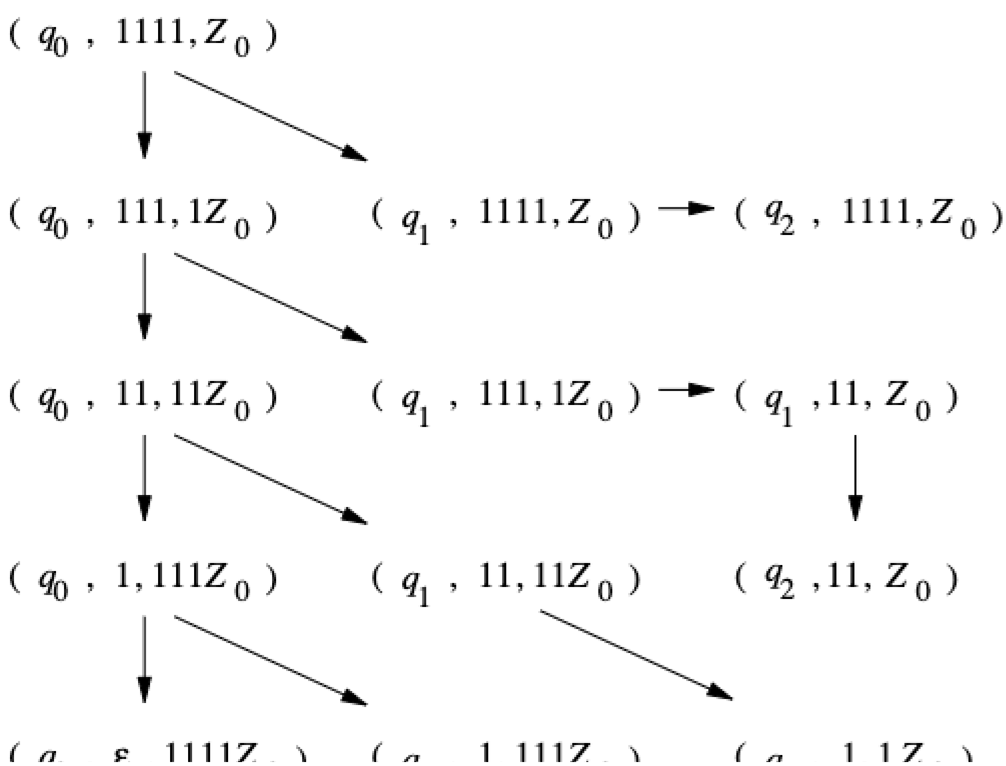
- a $\rightarrow$ primo simbolo dell'input
- X $\rightarrow$ stringa che va nello stack
  ed infatti dopo la transizione si ha che l'automa può essere rappresentato dalla seguente tripla $(p, w, a\beta)$ dove:

- $p \rightarrow$ è il nuovo stato dell'automa

Esempio: Su input 1111 il PDA



ha le seguenti sequenze di computazioni:



$(q_1, \epsilon, 1111Z_0)$ $(q_1, \epsilon, 11Z_0)$ (q_1, ϵ, Z_0)

39

(q_2, ϵ, Z_0)

Accettazione per Stato Finale

Sia $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ un PDA. Il *linguaggio accettato da P per stato finale* e'

$$L(P) = \{w : (q_0, w, Z_0) \overset{*}{\vdash} (q, \epsilon, \alpha),\ q \in F\}.$$

Esempio: Il PDA di prima accetta esattamente L_{wwr}.

[14_10_2025]

Accettazione per Pila Vuota

Sia $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ un PDA. Il *linguaggio accettato da P per pila vuota* e'

$$N(P) = \{w : (q_0, w, Z_0) \overset{*}{\vdash} (q, \epsilon, \epsilon)\}.$$

Nota: q puo' essere uno stato qualunque.

Domanda: come modificare il PDA per ww^R per accettare lo stesso linguaggio per pila vuota?

> Adesso chiedo che la pila sia vuota e q può essere uno stato non di accettazione

Noi Vogliamo Arrivare a Questo (Parte Destra Delle Frecce, Da PDA a PDA)

Un linguaggio e'

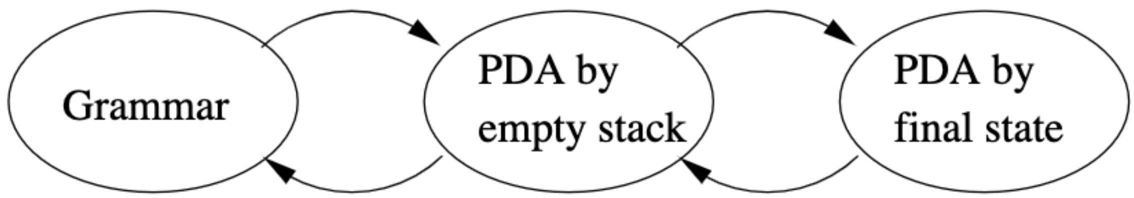
generato da una CFG

se e solo se e'

accettato da un PDA per pila vuota

se e solo se e'

accettato da un PDA per stato finale



Sappiamo gia' andare da pila vuota a stato finale.

Sono equipotenti, e possiamo passare da uno all'altro attraverso l'applicazione di un algoritmo.

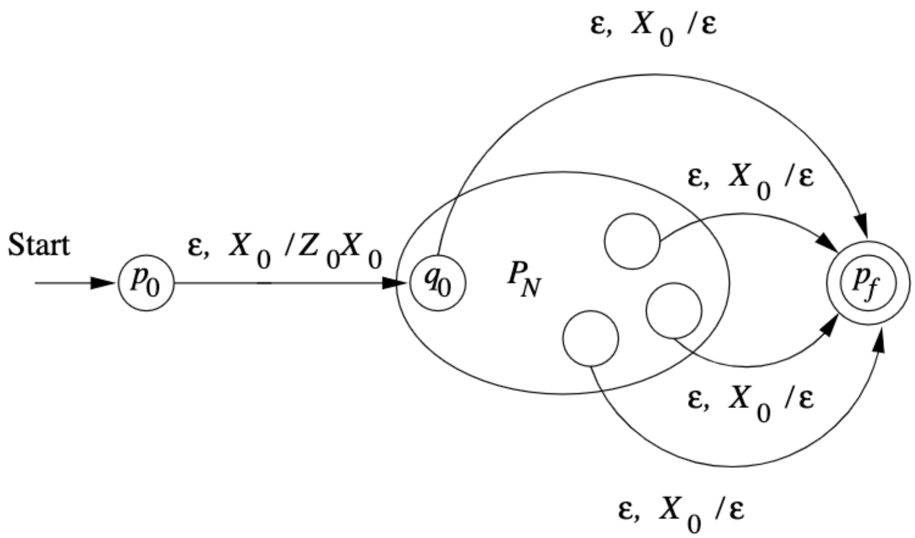
Da Pila Vuota a Stato Finale

Teorema 6.9: Se $L = N(P_N)$ per un PDA $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$, allora $\exists$ PDA P_F, tale che $L = L(P_F)$.

Prova: Sia

$$P_F = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$$

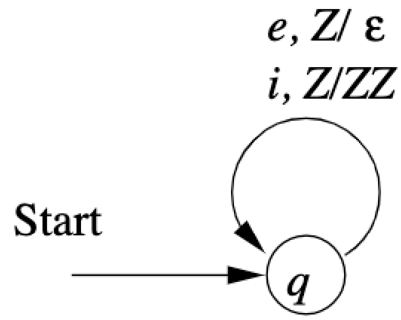
dove $\delta_F(p_0, \epsilon, X_0) = \{(q_0, Z_0X_0)\}$, e per ogni $q \in Q, a \in \Sigma \cup \{\epsilon\}, Y \in \Gamma : \delta_F(q, a, Y) = \delta_N(q, a, Y)$, e inoltre $(p_f, \epsilon) \in \delta_F(q, \epsilon, X_0)$.



Consiste in una emulazione di un PDA ad accettazione per Pila vuota su un PDA ad accettazione per Stato finale.
Di fatto prima di far partire il PDA "emulato" si aggiunge una X_0 nuovo, poi si fa esegue $\epsilon X_0/Z_0X_0$ che rende la pila così:

- posizione 0 $\rightarrow X_0$
- posizione 1 (next pop) $\rightarrow Z_0$
  partire il PDA emulato, quando egli finisce ad ogni stato parto una transizione che chiede che ci sia X_0 nella pila e porta il PDA da pila vuota allo stato finale del PDA a stato finale(padre)

Consideriamo il seguente automa a pila:



Formalmente,

$$P_N = (\{q\}, \{i, e\}, \{Z\}, \delta_N, q, Z),$$

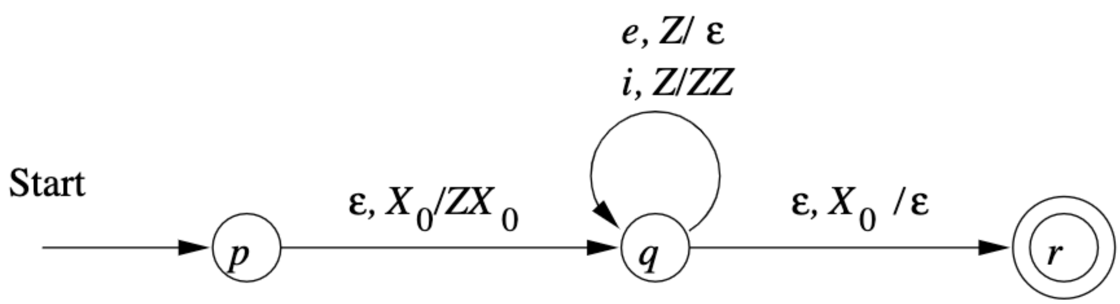
dove $\delta_N(q, i, Z) = \{(q, ZZ)\}$, e $\delta_N(q, e, Z) = \{(q, \epsilon)\}$.

Da P_N possiamo costruire

$$P_F = (\{p, q, r\}, \{i, e\}, \{Z, X_0\}, \delta_F, p, X_0, \{r\}),$$

dove
$\delta_F(p, \epsilon, X_0) = \{(q, ZX_0)\}$,
$\delta_F(q, i, Z) = \delta_N(q, i, Z) = \{(q, ZZ)\}$,
$\delta_F(q, e, Z) = \delta_N(q, e, Z) = \{(q, \epsilon)\}$, and
$\delta_F(q, \epsilon, X_0) = \{(r, \epsilon)\}$

Il diagramma per P_F e'



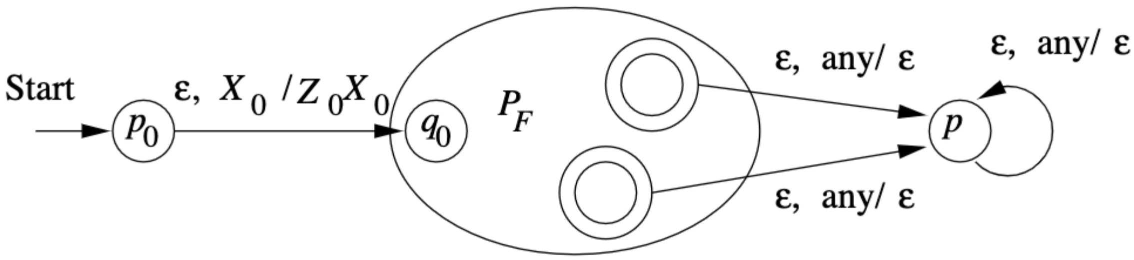
Da Stato Finale a Pila Vuota

Teorema 6.11: Sia $L = L(P_F)$, per un PDA $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$ Allora $\exists$ PDA P_N, tale che $L = N(P_N)$.

Prova: Sia

$$P_N = (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)$$

dove $\delta_N(p_0, \epsilon, X_0) = \{(q_0, Z_0X_0)\}$, $\delta_N(p, \epsilon, Y) = \{(p, \epsilon)\}$, per $Y \in \Gamma \cup \{X_0\}$, e per tutti i $q \in Q$, $a \in \Sigma \cup \{\epsilon\}, Y \in \Gamma : \delta_N(q, a, Y) = \delta_F(q, a, Y)$, e inoltre $\forall q \in F$, e $Y \in \Gamma \cup \{X_0\} : (p, \epsilon) \in \delta_N(q, \epsilon, Y)$.



Qua invece si mettono delle transizioni ad ogni stato del PDA a stato finale che porta ad uno stato specifico (svuotatore) che cicla fino a quando la pila non è vuota, e per evitare che la pila si svuoti a caso (cosa che non vogliamo se no succede il delirio) facciamo come prima, quindi ci mettiamo un X_0.

Adesso Faccio la Parte Sinistra Delle Frecce (Da PDA a Grammatica)

Idea: data G, costruiamo un PDA che simula $\overset{*}{\underset{lm}{\Rightarrow}}$.

Scriviamo le stringhe ottenute lungo una *derivazione sinistra* come

$$xA\alpha$$

dove A e' la variabile *piu' a sinistra*. Ad esempio,

$$\underbrace{\underbrace{(a+}_{x}\underbrace{E}_{A}\underbrace{)}_{\alpha}}$$

$$\underbrace{E\,)}_{\text{tail}}$$

Sia $xA\alpha \underset{lm}{\Rightarrow} x\beta\alpha$ (a causa di una produzione $A \to \beta$ della CFG). Questo corrisponde al PDA che, dopo aver consumato input x, e essersi ritrovato con $A\alpha$ sulla pila, ora esegue una transizione ϵ che elimina A e mette al suo posto β sulla pila.

Piu' formalmente, sia w la stringa data in *input* al PDA e y tale che $w = xy$. Allora il PDA va non deterministicamente dalla configurazione $(q, y, A\alpha)$ alla configurazione $(q, y, \beta\alpha)$.

Left-Most.

Lui considera la variabile più a sinistra e viene messa nello stack al next pop.

Alla configurazione $(q, y, \beta\alpha)$ il PDA si comporta come prima, a meno che ci siano *terminali* nel prefisso di β. In questo caso, il PDA li elimina, se *li legge nell'input* (se fanno match con l'input).

Se tutte le scommesse sono giuste (consentono di matchare l'input), il PDA finisce l'input con la *pila vuota*.

Quindi **la trasformazione è la seguente**.

Sia $G = (V, T, Q, S)$ una CFG. Definiamo P_G come

$$(\{q\}, T, V \cup T, \delta, q, S),$$

dove

$$\delta(q, \epsilon, A) = \{(q, \beta) : A \to \beta \in Q\},$$

per $A \in V$, e

$$\delta(q, a, a) = \{(q, \epsilon)\},$$

per $a \in T$.

Esempio:

Consideriamo la grammatica

$$S \to \epsilon|SS|iS|iSe.$$

Il PDA corrispondente e'

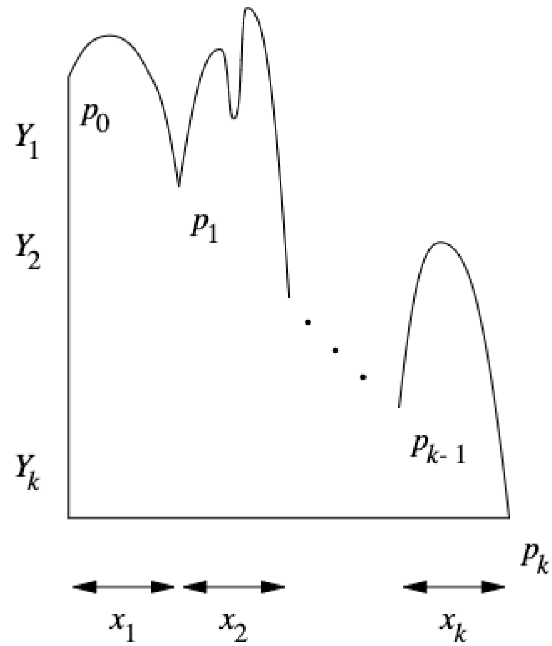
$$P = (\{q\}, \{i, e\}, \{S, i, e\}, \delta, q, S),$$

dove $\delta(q, \epsilon, S) = \{(q, \epsilon), (q, SS), (q, iS), (q, iSe)\}$, $\delta(q, i, i) = \{(q, \epsilon)\}$, e $\delta(q, e, e) = \{(q, \epsilon)\}$.

15/10/2025 Fino a Pag

Da PDA a CFG

Idea: comportamento dei PDA per rimuovere simbolo Y dalla pila (usando una transizione che sostituisce Y con $Y_1 Y_2 \cdots Y_k$)



Definiremo una grammatica con variabili della forma $[p_{i-1}Y_i p_i]$ che rappresentano il passaggio da p_{i-1} a p_i con l'effetto di eliminare Y_i.

Quindi stringa terminale generata da variabile $[pXq]$ rappresenta: **input letto da PDA andando da p a q e rimuovendo X da pila**

Formalmente, sia $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ un PDA. Definiamo $G = (V, \Sigma, R, S)$, con

$$
\begin{aligned}
V &= \{[pXq] : \{p, q\} \subseteq Q, X \in \Gamma\} \cup \{S\} \\
R &= \{S \to [q_0 Z_0 p] : p \in Q\} \cup \\
&\quad \{[\mathbf{q} X r_k] \to a[\mathbf{r} Y_1 r_1] \cdots [r_{k-1} Y_k r_k] : \\
&\qquad\qquad a \in \Sigma \cup \{\epsilon\}, \\
&\qquad\qquad \{r_1, \ldots, r_k\} \subseteq Q, \\
&\qquad\qquad (\mathbf{r}, Y_1 Y_2 \cdots Y_k) \in \delta(\mathbf{q}, a, X)\}
\end{aligned}
$$

dove, in caso $k = 0$ si ha: $Y_1 Y_2 \cdots Y_k = \epsilon$ e $r_k = \mathbf{r}$

Esempio: Convertiamo $P = (\{p,q\},\{0,1\},\{X,Z_0\},\delta,q,Z_0)$, dove δ e' data da

1. $\delta(q,1,Z_0) = \{(q,XZ_0)\}$

2. $\delta(q,1,X) = \{(q,XX)\}$

3. $\delta(q,0,X) = \{(p,X)\}$

4. $\delta(q,\epsilon,X) = \{(q,\epsilon)\}$

5. $\delta(p,1,X) = \{(p,\epsilon)\}$

6. $\delta(p,0,Z_0) = \{(q,Z_0)\}$

in una CFG.

Otteniamo $G = (V,\{0,1\},R,S)$, dove

$$V = \{[qZ_0q],[pZ_0q],[qZ_0p],[pZ_0p],[qXq],[pXq],[qXp],[pXp],S\}$$

e le produzioni in R sono

$S \to [qZ_0q]|[qZ_0p]$

Dalla transizione (1) $\delta(q,1,Z_0) = \{(q,XZ_0)\}$ si ha:

$[qZ_0q] \to 1[qXq][qZ_0q]$
$[qZ_0q] \to 1[qXp][pZ_0q]$
$[qZ_0p] \to 1[qXq][qZ_0p]$
$[qZ_0p] \to 1[qXp][pZ_0p]$

Dalla transizione (2) $\delta(q,1,X) = \{(q,XX)\}$ si ha:

$[qXq] \to 1[qXq][qXq]$
$[qXq] \to 1[qXp][pXq]$
$[qXp] \to 1[qXq][qXp]$
$[qXp] \to 1[qXp][pXp]$

Dalla transizione (3) $\delta(q, 0, X) = \{(p, X)\}$ si ha:

$[qXq] \rightarrow 0[pXq]$
$[qXp] \rightarrow 0[pXp]$

Dalla transizione (4) $\delta(q, \epsilon, X) = \{(q, \epsilon)\}$ si ha:

$[qXq] \rightarrow \epsilon$

Dalla transizione (5) $\delta(p, 1, X) = \{(p, \epsilon)\}$ si ha:

$[pXp] \rightarrow 1$

Dalla transizione (6) $\delta(p, 0, Z_0) = \{(q, Z_0)\}$ si ha:

$[pZ_0q] \rightarrow 0[qZ_0q]$
$[pZ_0p] \rightarrow 0[qZ_0p]$

PDA Deterministici

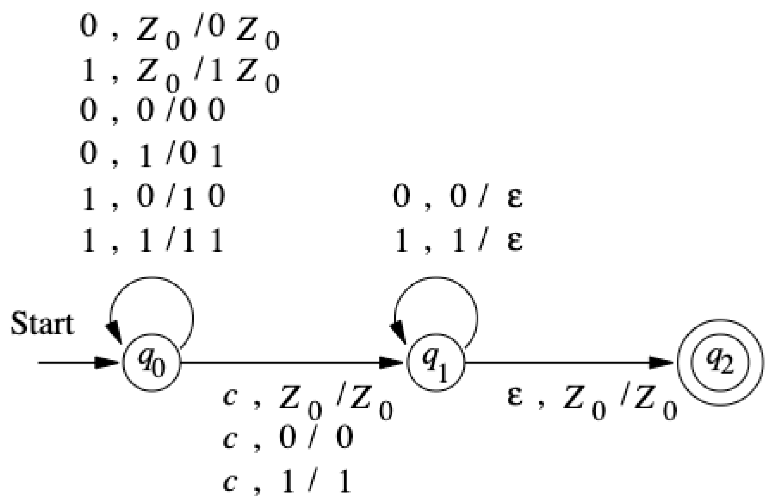
Un PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ e' *deterministico* se e solo se:

1. ogni $\delta(q, a, X)$, con $a \in \Sigma \cup \{\epsilon\}$, contiene *al piu'* un elemento
2. se $\delta(q, a, X)$ non vuoto per un $a \in \Sigma$, allora $\delta(q, \epsilon, X)$ vuoto.

Esempio: Definiamo

$$L_{wcwr} = \{wcw^R : w \in \{0, 1\}^*\}$$

Allora L_{wcwr} e' riconosciuto dal seguente DPDA



DPDA Che Accettano per Stato Finale

Mostreremo che Regolari $\subset L$(DPDA) $\subset$ CFL

Teorema 6.17: Se L e' regolare, allora $L = L(P)$ per qualche DPDA P.

Prova: Dato che L e' regolare, esiste un DFA A tale che $L = L(A)$. Sia

$$A = (Q, \Sigma, \delta_A, q_0, F)$$

definiamo il DPDA

$$P = (Q, \Sigma, \{Z_0\}, \delta_P, q_0, Z_0, F),$$

dove

$$\delta_P(q, a, Z_0) = \{(\delta_A(q, a), Z_0)\},$$

per tutti i $p, q \in Q$ e $a \in \Sigma$.

Un'induzione su $|w|$ ci da'

$$(q_0, w, Z_0) \overset{*}{\vdash} (p, \epsilon, Z_0) \Leftrightarrow \hat{\delta}_A(q_0, w) = p$$

- Abbiamo visto che Regolari $\subseteq L$(DPDA).
- $L_{wcwr} \in L$(DPDA)$\setminus$ Regolari
- Ci sono linguaggi in CFL$\setminus L$(DPDA).

  Si, per esempio L_{wwr}.

DPDA Che Accettano per Pila Vuota

E i DPDA che accettano per pila vuota?

Possono riconoscere solo linguaggi con la **proprieta' del prefisso**.

Un linguaggio L ha la *proprieta' del prefisso* se **non** esistono due stringhe distinte in L, tali che una e' un prefisso dell'altra.

Esempio: L_{wcwr} ha la proprieta' del prefisso.

Esempio: $\{0\}^*$ non ha la proprieta' del prefisso.

Teorema 6.19: L e' $N(P)$ per qualche DPDA P se e solo se L ha la proprieta' del prefisso e L e' $L(P')$ per qualche DPDA P'.

DPDA E Non Ambiguità

L(DPDA) coincide con i CFL aventi grammatiche **non ambigue** (cioe' non inerentemente ambigui)? **No**. Per esempio:

> L_{wwr} ha una grammatica non ambigua $S \to 0S0|1S1|\epsilon$ ma non e' L(DPDA).

L'inverso invece vale! Abbiamo, preliminarmente:

Teorema 6.20: Se $L = N(P)$ per qualche DPDA P, allora L ha una CFG non ambigua.

Prova: Applicando la costruzione vista da PDA a CFG, se la costruzione e' applicata ad un DPDA, il risultato e' una CFG con derivazioni a sinistra uniche per ogni stringa.

Teorema 6.20 puo' essere rafforzato:

Teorema 6.21: Se $L = L(P)$ per qualche DPDA P, allora L ha una CFG non ambigua.

Prova: Sia $ un simbolo fuori dell'alfabeto di L, e sia $L' = L\{\$\}$.
E' facile modificare P per riconoscere L' (PDA ancora deterministico); inoltre L' ha la proprieta' del prefisso.
Per il teorema 6.19 abbiamo $L' = N(P')$ per qualche DPDA P'.
Per il teorema 6.20 L' puo' essere generato da una CFG G' non ambigua
Modifichiamo G' in G, tale che $L(G) = L$, aggiungendo la produzione

$$\$ \to \epsilon$$

(e considerando $ una variabile anziche' un terminale)

Dato che G' ha derivazioni a sinistra uniche, anche G le avra' uniche, dato che l'unica cosa nuova e' l'aggiunta di derivazioni

$$w\$ \underset{lm}{\Rightarrow} w$$

alla fine.

21/10/2025 - Continuo Da Pagina 64

Proprietà Dei CFL

- *Semplificazione* di una CFG. Se un linguaggio e' un CFL, ha una grammatica in una possibile forma speciale.

- *Pumping Lemma per CFL.* Simile ai linguaggi regolari.

- *Proprieta' di chiusura.* Solo alcune delle proprieta' di chiusura dei linguaggi regolari valgono anche per i CFL.

- *Proprieta' di decisione.* Possiamo controllare l'appartenenza e l'essere vuoto, ma, per esempio, l'equivalenza di CFL e' non verificabile tramite un algoritmo (indecidibile).

Forma Normale Di Chomsky

Ogni CFL (senza ϵ) e' generato da una CFG dove tutte le produzioni sono della forma

$$A \to BC, \text{ o } A \to a$$

dove A, B, e C sono variabili, e a e' un simbolo terminale. Questa e' detta forma normale di Chomsky (CNF), e per ottenerla dobbiamo innanzitutto "pulire" la grammatica:

- Eliminare i *simboli inutili*, quelli che non appaiono in nessuna derivazione $S \stackrel{*}{\Rightarrow} w$, per simbolo iniziale S e terminale w.

- Eliminare le produzioni ϵ, della forma $A \to \epsilon$.

- Eliminare le *produzioni unita'*, cioe' produzioni della forma $A \to B$, dove A e B sono variabili.

l'albero sintattico della grammatica è binario.
Perderemo la stringa vuota se seguiamo questa forma normale di Chomsky, perchè non possiamo più rappresentare ϵ.

Eliminazione Simboli Inutili

- Un simbolo X e' *utile* per una grammatica $G = (V, T, P, S)$, se esiste una derivazione

$$S \overset{*}{\underset{G}{\Rightarrow}} \alpha X \beta \overset{*}{\underset{G}{\Rightarrow}} w$$

per una stringa di terminali w. Simboli che non sono utili sono detti *inutili*.

- Un simbolo X e' *generante* se $X \overset{*}{\underset{G}{\Rightarrow}} w$, per qualche $w \in T^*$

- Un simbolo X e' *raggiungibile* se $S \overset{*}{\underset{G}{\Rightarrow}} \alpha X \beta$, per qualche $\{\alpha, \beta\} \subseteq (V \cup T)^*$

Se in G (con $L(G) \neq \emptyset$) eliminiamo prima i simboli non generanti, e poi quelli non raggiungibili, rimarranno solamente simboli utili.

Esempio

Esempio: Sia G la grammatica

$$S \to AB|a, \ A \to b$$

S e A sono generanti, B non lo e'. Se eliminiamo B dobbiamo eliminare $S \to AB$, riducendo la grammatica

$$S \to a, \ A \to b$$

Ora, solo la variabile S e' raggiungibile. Eliminando A rimane solo

$$S \to a$$

con linguaggio $\{a\}$.

Nota Se eliminiamo prima i simboli non raggiungibili, si ha che tutti i simboli sono raggiungibili. Da

$$S \to AB|a, \ A \to b$$

eliminiamo B in quanto non generante, e rimane la grammatica

$$S \to a, \ A \to b$$

che contiene ancora simboli inutili

Eliminazione Produzioni ϵ

Si ha che se L e' un CFL, allora $L \setminus \{\epsilon\}$ ha una grammatica priva di produzioni ϵ (cio' mostra anche che $L \setminus \{\epsilon\}$ e' CFL).

La variabile A e' *annullabile* se $A \stackrel{*}{\Rightarrow} \epsilon$.

Sia A annullabile. Rimpiazzeremo una regola del tipo

$$B \rightarrow \alpha A \beta$$

con

$$B \rightarrow \alpha A \beta, \ B \rightarrow \alpha \beta$$

(rimpiazzando in tal modo anche le nuove regole via via ottenute) e cancelleremo tutte le regole con corpo ϵ.

Indichiamo con $n(G)$, l'insieme dei simboli annullabili di una grammatica $G = (V, T, P, S)$

Esempio

Esempio: Sia G la grammatica

$$S \rightarrow AB, \ A \rightarrow aAA|\epsilon, \ B \rightarrow bBB|\epsilon$$

Abbiamo $n(G) = \{A, B, S\}$. La prima regola diventa

$$S \rightarrow AB|A|B$$

la seconda

$$A \rightarrow aAA|aA|aA|a$$

e la terza

$$B \rightarrow bBB|bB|bB|b$$

Eliminiamo le regole con corpo ϵ, ed otteniamo la grammatica G_1 :

$$S \rightarrow AB|A|B, \ A \rightarrow aAA|aA|a, \ B \rightarrow bBB|bB|b$$

Eliminazione Produzione Unità

$$A \to B$$

e' una produzione *unita'*, nel caso in cui A e B siano variabili.

Produzioni unita' possono essere eliminate.

Si consideri la grammatica

$$\begin{aligned} E &\to T \mid E + T \\ T &\to F \mid T * F \\ F &\to I \mid (E) \\ I &\to a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \end{aligned}$$

ha le produzioni unita' $E \to T$, $T \to F$, e $F \to I$

Si consideri la **produzione unità** $\mathbf{E} \to \mathbf{T}$. Si trasforma tale produzione con il seguente procedimento a **espansione**.

Si espande $E \to T$ ottenendo le produzioni:

$$E \to F,\ E \to T * F$$

Poi, espandendo $E \to F$, si ottiene:

$$E \to I|(E)|T * F$$

Infine, espandendo $E \to I$, si ottiene:

$$E \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \mid (E) \mid T * F$$

Si considerano poi le altre produzioni unità $\mathbf{T} \to \mathbf{F}$ e $\mathbf{F} \to \mathbf{I}$ della grammatica e, per ciascuna, si applica analogo procedimento.

La grammatica inziale

$$
\begin{aligned}
E &\rightarrow T \mid E+T \\
T &\rightarrow F \mid T*F \\
F &\rightarrow I \mid (E) \\
I &\rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1
\end{aligned}
$$

viene quindi modificata trasformando:

$E \rightarrow T$ in

$$E \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \mid (E) \mid T*F$$

$T \rightarrow F$ in

$$T \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \mid (E)$$

$F \rightarrow I$ in

$$F \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$$

Quindi, eliminando le produzioni unità, la grammatica diviene

$$
\begin{aligned}
E &\rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \mid (E) \mid T*F \mid E+T \\
T &\rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \mid (E) \mid T*F \\
F &\rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \mid (E) \\
I &\rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1
\end{aligned}
$$

PROBLEMA - Presenza di cicli

Questa trasformazione non va bene nei linguaggi che rappresentano cicli, bisogna fare un accorgimento:
Se incontro una produzione che ho già espanso, posso fermarmi ed

eliminarla come di seguito:

Esempio: si consideri la grammatica

$A \rightarrow B \mid a$
$B \rightarrow C \mid b$
$C \rightarrow A \mid c$

Comincio trasformando la **produzione unità $\mathbf{A \rightarrow B}$**.

Si espande $A \rightarrow B$ ottenendo le produzioni:

$$A \rightarrow C \mid b$$

Poi, espandendo $A \rightarrow C$, si ottiene:

$$A \rightarrow A \mid c \mid b$$

Infine, espandendo $A \rightarrow A$, si ottiene:

$$A \rightarrow B \mid a \mid c \mid b$$

Andando avanti ottengo ovviamente sempre produzioni che ho già, quindi mi posso **fermare ed eliminare $\mathbf{A \rightarrow B}$**.

La produzione unità $A \rightarrow B$ si trasforma quindi in:

$$A \rightarrow a \mid c \mid b$$

Forma Normale di Chomsky per CNF

Ogni CFL non vuoto, che non contiene ϵ, ha una grammatica G priva di simboli inutili, con produzioni nella forma

- $A \to BC$, dove $\{A, B, C\} \subseteq V$, o
- $A \to a$, dove $A \in V$, e $a \in T$.

Per ottenerla, si effettuano le seguenti trasformazioni su una qualsiasi grammatica per il CFL

1. "Pulire" la grammatica
2. Modificare le produzioni con 2 o piu' simboli in modo tale che siano tutte variabili
3. Ridurre il corpo delle regole di lunghezza superiore a 2 in cascate di produzioni con corpi da 2 variabili.

• Per il passo 2, per ogni terminale a che compare in un corpo di lunghezza ≥ 2, creare una nuova variabile, ad esempio A, e sostituire a con A in tutti i corpi, e aggiungere la nuova regola $A \to a$.

• Per il passo 3, per ogni regola nella forma

$$A \to B_1 B_2 \cdots B_k,$$

$k \geq 3$, introdurre le nuove variabili $C_1, C_2, \ldots C_{k-2}$, e sostituire la regola con

$$\begin{array}{rcl} A & \to & B_1 C_1 \\ C_1 & \to & B_2 C_2 \\ & \cdots & \\ C_{k-3} & \to & B_{k-2} C_{k-2} \\ C_{k-2} & \to & B_{k-1} B_k \end{array}$$