Capítulo 3: Autômatos de Pilha

Newton José Vieira

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais

16 de setembro de 2010

Sumário

Sumário

Sumário

Sumário

Linguagens não regulares

Exemplo de linguagem **não regular** muito comum:

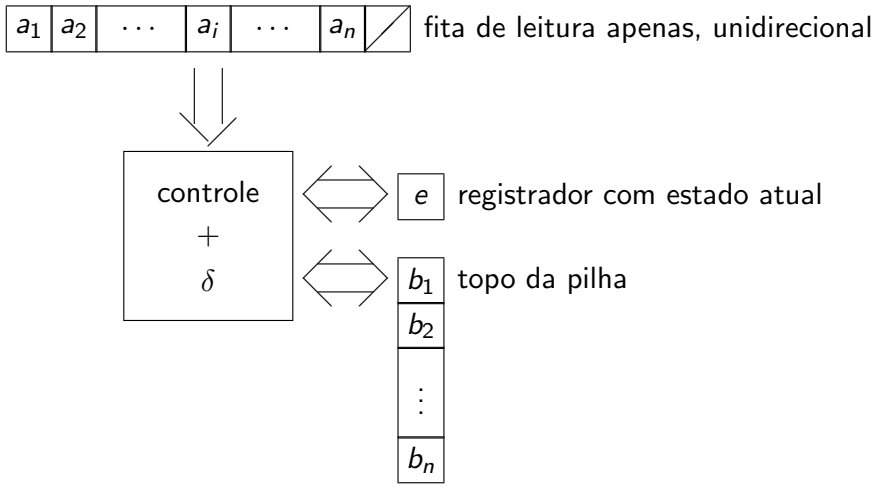
Linguagem que contém expressões aritméticas, como:

$$({}^{n}t_1+t_2)+t_3)\cdots+t_{n+1})$$

em que $n \geq 1$, cada t_i é uma subexpressão, e o número de (s é igual ao de)s.

Intuitivamente, um AF não pode "lembrar" que leu n ocorrências de certo símbolo, para n **arbitrário**.

Arquitetuta de um autômato de pilha



Transição de um AP

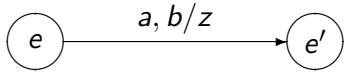
Sejam:

- E: conjunto de estados;
- Σ: alfabeto de entrada;
- Γ: alfabeto de pilha;

Cada transição do AP é da forma

$$\delta(e, a, b) = [e', z]$$

$e, e' \in E$, $a \in \Sigma \cup \{\lambda\}$, $b \in \Gamma \cup \{\lambda\}$ e $z \in \Gamma^*$.
Essa transição em um diagrama de estados:

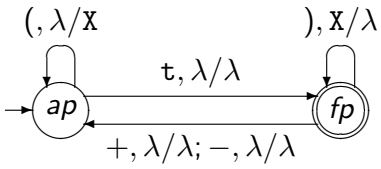


Um exemplo de AP

Conjunto EA das expressões aritméticas:

a) $\mathtt{t} \in \text{EA}$;

b) se $x, y \in \text{EA}$, então $(x) \in \text{EA}$, $x{+}y \in \text{EA}$ e $x{-}y \in \text{EA}$.



Alfabetos:

- $\Sigma = \{\mathtt{t}, (,), +, -\}$;
- $\Gamma = \{\mathtt{X}\}$;

Computação de um AP

- Pilha: uma palavra de Γ^*. Pilha vazia: λ.
- Configuração instantânea: $[e, y, p]$, sendo p a pilha.

Computação do AP quando a palavra de entrada é `(t-(t+t))`:

$$\begin{aligned}[ap, (\mathtt{t} - (\mathtt{t} + \mathtt{t})), \lambda] &\vdash [ap, \mathtt{t} - (\mathtt{t} + \mathtt{t})), X] \\ &\vdash [fp, -(\mathtt{t} + \mathtt{t})), X] \\ &\vdash [ap, (\mathtt{t} + \mathtt{t})), X] \\ &\vdash [ap, \mathtt{t} + \mathtt{t})), XX] \\ &\vdash [fp, +\mathtt{t})), XX] \\ &\vdash [ap, \mathtt{t})), XX] \\ &\vdash [fp,)), XX] \\ &\vdash [fp,), X] \\ &\vdash [fp, \lambda, \lambda].\end{aligned}$$

Outros exemplos/condições para reconhecimento

$$[ap, \mathtt{t}), \lambda] \vdash [fp,), \lambda].$$

$\Longrightarrow$ o AP para **sem consumir toda a palavra** de entrada.

$$\begin{aligned}[ap, (\mathtt{t}, \lambda] &\vdash [ap, \mathtt{t}, \mathtt{X}] \\ &\vdash [fp, \lambda, \mathtt{X}].\end{aligned}$$

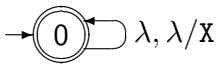
$\Longrightarrow$ o AP para em estado final com **pilha não vazia**.

Para uma palavra ser reconhecida:

- ela deve ser totalmente consumida;
- a máquina deve terminar em um estado final;
- a deve estar pilha vazia.

Um exemplo estranho

Seja o AP com $\Sigma = \{1\}$ e com o diagrama de estados:



Para toda palavra em $\{1\}^+$, o AP não para. Em particular:

$$[0, 1, \lambda] \vdash [0, 1, X] \vdash [0, 1, XX] \ldots$$

Perguntas:

- para λ, o AP para ou não?
- λ é reconhecida ou não?

Transições compatíveis

Transições compatíveis

Seja $\delta : E \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \to E \times \Gamma^*$.

As transições $\delta(e, a, b)$ e $\delta(e, a', b')$ são compatíveis

se, e somente se

$(a = a'$ ou $a = \lambda$ ou $a' = \lambda)$ e $(b = b'$ ou $b = \lambda$ ou $b' = \lambda)$.

Ou ainda: $\delta(e, a, b)$ e $\delta(e, a', b')$ são **incompatíveis** se, e somente se:

$$(a \neq a' \text{ e } a \neq \lambda \text{ e } a' \neq \lambda) \text{ ou } (b \neq b' \text{ e } b \neq \lambda \text{ e } b' \neq \lambda).$$

O que é AP determinístico

Autômato de pilha determinístico

Um autômato de pilha determinístico *(APD)* é uma sêxtupla $(E, \Sigma, \Gamma, \delta, i, F)$, em que

- E é um conjunto finito de um ou mais estados;
- Σ é o alfabeto de entrada;
- Γ é o alfabeto de pilha;
- δ é uma função parcial de $E \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\})$ para $E \times \Gamma^*$, sem transições compatíveis;
- $i \subseteq E$ é o estado inicial;
- $F \subseteq E$ é o conjunto de estados finais.

A linguagem reconhecida por um APD

Seja um APD $M = (E, \Sigma, \Gamma, \delta, i, F)$. Então:
$\forall e, e' \in E$, $a \in \Sigma \cup \{\lambda\}$, $b \in \Gamma \cup \{\lambda\}$, $x \in \Gamma^*$:

$$\forall y \in \Sigma^*, z \in \Gamma^* [e, ay, bz] \vdash [e', y, xz] \leftrightarrow \delta(e, a, b) = [e', x].$$

$\overset{*}{\vdash}$ é o fecho reflexivo e transitivo de $\vdash$.

Linguagem reconhecida por um APD

Seja $M = (E, \Sigma, \Gamma, \delta, i, F)$. A linguagem reconhecida por M é:

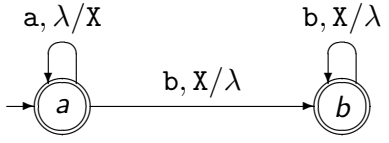
$$L(M) = \{w \in \Sigma^* \mid [i, w, \lambda] \overset{*}{\vdash} [e, \lambda, \lambda] \text{ para algum } e \in F\}.$$

Exemplo

$\{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbf{N}\}$ é reconhecida por $(\{a, b\}, \{\mathtt{a}, \mathtt{b}\}, \{\mathtt{X}\}, \delta, a, \{a, b\})$, em que δ é dada por:

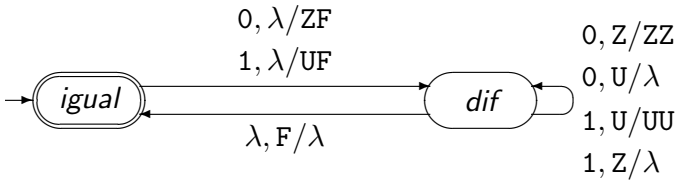
1. $\delta(a, \mathtt{a}, \lambda) = [a, \mathtt{X}]$;
2. $\delta(a, \mathtt{b}, \mathtt{X}) = [b, \lambda]$;
3. $\delta(b, \mathtt{b}, \mathtt{X}) = [b, \lambda]$.

Diagrama de estados:



Outro exemplo

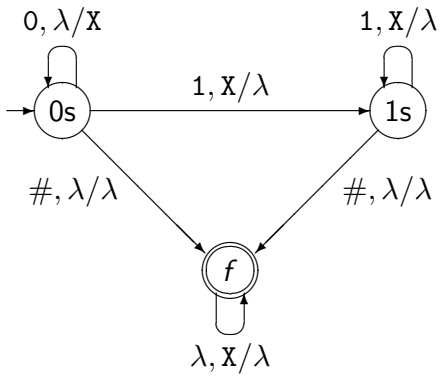
APD que reconhece
$\{w \in \{0,1\}^* \mid$ o número de 0s em w é igual ao de 1s$\}$:



$\Longrightarrow$ Este APD utiliza uma técnica para **verificar se a pilha chegou ao fundo**.

Mais um exemplo

APD que reconhece $\{0^m 1^n \# \mid m \geq n\}$:



$\{0^m 1^n \mid m \geq n\}$ não pode ser reconhecida por APD!

Algoritmo para APD

Entrada: (1) o APD, dado por i, F e D, e
(2) a palavra de entrada, dada por *prox*.
Saída: *sim* ou *não*.
$e \leftarrow i$; *empilhe*($\bigtriangledown$); $ps \leftarrow$ *prox*();
enquanto $D[e, a, b]$ é definido p/ $a \in \{ps, \lambda\}$ e $b \in \{topo(), \lambda\}$ **faça**
  seja $D[e, a, b] = [e', z]$;
  se $a \neq \lambda$ **então** $ps \leftarrow$ *prox*() **fimentão;**
  se $b \neq \lambda$ **então** *desempilhe*() **fimentão;**
  empilhe(z);
  $e \leftarrow e'$
fimenquanto;
se $ps = fs$ e $topo() = \bigtriangledown$ e $e \in F$ **então**
  retorne *sim*
senão
  retorne *não*
fimse

Esse "algoritmo" pode entrar em loop!

O que é AP não determinístico

Autômato de pilha não determinístico

Um autômato de pilha não determinístico *(APN)* é uma sêxtupla $(E, \Sigma, \Gamma, \delta, I, F)$, em que

- E, Σ, Γ e F são como em APDs;
- δ é uma função parcial de $E \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\})$ para D, sendo D constituído dos subconjuntos finitos de $E \times \Gamma^*$;
- $I \subseteq E$ é o conjunto de estados iniciais.

A linguagem reconhecida por um APN

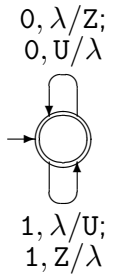
Linguagem reconhecida por um APN

Seja um APN $M = (E, \Sigma, \Gamma, \delta, I, F)$. A *linguagem reconhecida* por M é:

$$L(M) = \{w \in \Sigma^* \mid [i, w, \lambda] \overset{*}{\vdash} [e, \lambda, \lambda] \text{ para algum } i \in I \text{ e } e \in F\}.$$

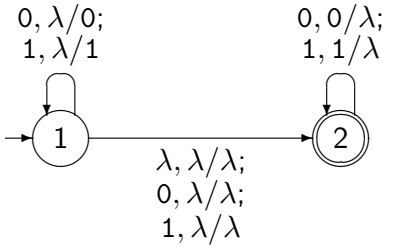
Exemplo de APN

APD que reconhece
$\{w \in \{0,1\}^* \mid$ o número de 0s em w é igual ao de 1s$\}$:



Exemplo não tratável por APD

APN que reconhece a linguagem $\{w \in \{0,1\}^* \mid w = w^R\}$:



Em uma computação de sucesso para w:

- se $|w|$ for par, será percorrida a transição de 1 para 2 sob λ;
- se $|w|$ for ímpar e o símbolo do meio for a, será percorrida a transição de 1 para 2 sob a.

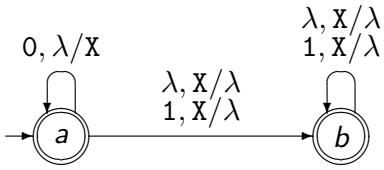
Reconhecimento por estado final

Reconhecimento por estado final

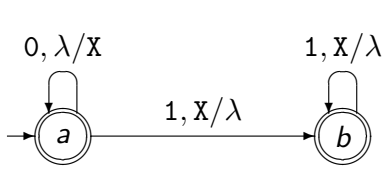
Seja um APN $M = (E, \Sigma, \Gamma, \delta, I, F)$. A linguagem reconhecida por M por estado final é:

$$L_F(M) = \{w \in \Sigma^* \mid [i, w, \lambda] \overset{*}{\vdash} [e, \lambda, y] \text{ para } i \in I, e \in F \text{ e } y \in \Gamma^*\}.$$

Exemplo: APNs para $L = \{0^m 1^n \mid m \geq n\}$:



Aceitação por pilha vazia e estado final.



Aceitação por estado final

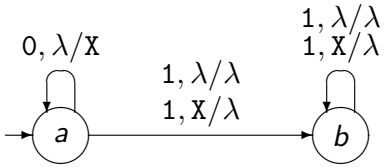
Reconhecimento por pilha vazia

Reconhecimento por pilha vazia

Seja um APN $M = (E, \Sigma, \Gamma, \delta, I)$. A linguagem reconhecida por M por pilha vazia é:

$L_V(M) = \{w \in \Sigma^* \mid [i, w, \lambda] \overset{*}{\vdash} [e, \lambda, \lambda] \text{ para algum } i \in I \text{ e } e \in E\}$.

Exemplo: APN para $\{0^m 1^n \mid m \leq n\}$, rec. por pilha vazia:



Equivalência de métodos de reconhecimento

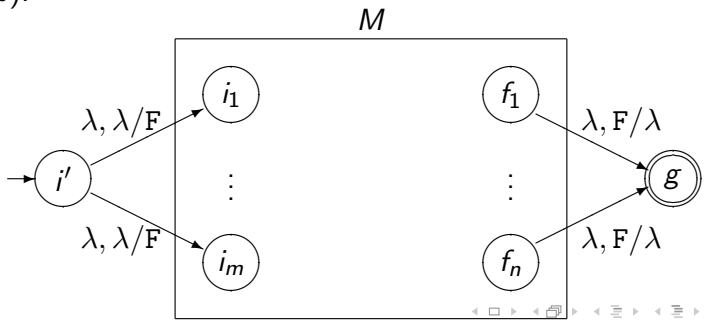
Seja L uma linguagem. As seguinte afirmativas são equivalentes:

a) L pode ser reconhecida por pilha vazia e estado final.

b) L pode ser reconhecida por estado final.

c) $L \cup \{\lambda\}$ pode ser reconhecida por pilha vazia.

(a) $\rightarrow$ (b):



Equivalência de métodos de reconhecimento

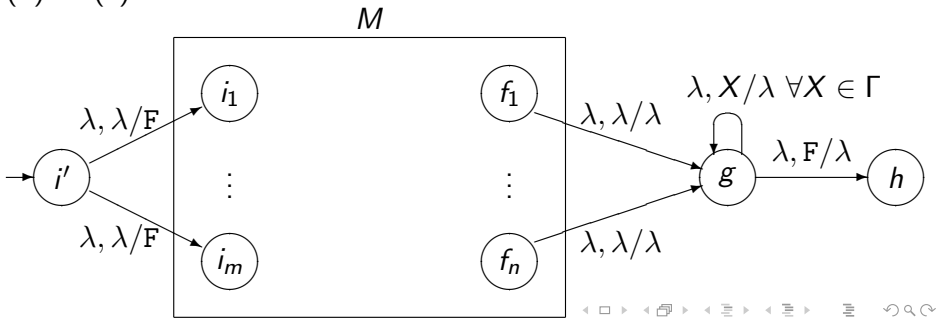
Seja L uma linguagem. As seguinte afirmativas são equivalentes:

a) L pode ser reconhecida por pilha vazia e estado final.

b) L pode ser reconhecida por estado final.

c) $L \cup \{\lambda\}$ pode ser reconhecida por pilha vazia.

(b) $\rightarrow$ (c)



Equivalência de métodos de reconhecimento

Seja L uma linguagem. As seguinte afirmativas são equivalentes:

a) L pode ser reconhecida por pilha vazia e estado final.

b) L pode ser reconhecida por estado final.

c) $L \cup \{\lambda\}$ pode ser reconhecida por pilha vazia.

(c) $\rightarrow$ (a)
Se $M = (E, \Sigma, \Gamma, \delta, I)$, então $M' = (E, \Sigma, \Gamma, \delta, I, E)$.

Gramática em notação BNF

$$
\begin{array}{rcl}
\langle \text{programa} \rangle & \to & \langle \text{declarações} \rangle \ ; \ \langle \text{lista-de-cmds} \rangle \ . \\
\vdots & & \\
\langle \text{lista-de-cmds} \rangle & \to & \langle \text{comando} \rangle \ ; \ \langle \text{lista-de-cmds} \rangle \mid \\
 & & \lambda \\
\langle \text{comando} \rangle & \to & \langle \text{cmd-enquanto} \rangle \mid \\
 & & \langle \text{cmd-se} \rangle \mid \\
 & & \langle \text{cmd-atribuição} \rangle \mid \\
 & & \ldots
\end{array}
$$

Gramática em notação BNF (cont.)

$$
\begin{array}{rcl}
\langle \text{cmd-enquanto} \rangle & \to & \textbf{enquanto} \ \langle \text{exp-lógica} \rangle \ \textbf{faça} \\
 & & \langle \text{lista-de-cmds} \rangle \ \textbf{fimenquanto} \\
\langle \text{cmd-se} \rangle & \to & \textbf{se} \ \langle \text{exp-lógica} \rangle \ \textbf{então} \\
 & & \langle \text{lista-de-cmds} \rangle \ \langle \text{senaoses} \rangle \ \langle \text{senao} \rangle \ \textbf{fimse} \\
\langle \text{senaoses} \rangle & \to & \textbf{senãose} \ \langle \text{exp-lógica} \rangle \ \textbf{então} \\
 & & \langle \text{lista-de-cmds} \rangle \ \langle \text{senaoses} \rangle \mid \\
 & & \lambda \\
\langle \text{senao} \rangle & \to & \textbf{senão} \ \langle \text{lista-de-cmds} \rangle \mid \\
 & & \lambda \\
\langle \text{cmd-atribuição} \rangle & \to & \langle \text{variável} \rangle \leftarrow \langle \text{expressão} \rangle
\end{array}
$$

O que é gramática livre do contexto

Gramática livre do contexto

Uma gramática livre do contexto (GLC) é uma gramática (V, Σ, R, P), em que cada regra tem a forma $X \to w$, em que $X \in V$ e $w \in (V \cup \Sigma)^*$.

Exemplo:

$\{0^n 1^n \mid n \in \mathbf{N}\}$ é gerada por $G = (\{P\}, \{0, 1\}, R, P)$, em que R consta das duas regras:

$P \to 0P1 \mid \lambda$

Mais exemplos de GLCs

GLC que gera $\{w \in \{0, 1\}^* \mid w = w^R\}$:

$G = (\{P\}, \{0, 1\}, R, P)$, tendo R as 5 regras:
$P \to 0P0 \mid 1P1 \mid 0 \mid 1 \mid \lambda$

GLC que gera $\{w \in \{0, 1\}^* \mid w$ tem um número igual de 0s e 1s$\}$:

$(\{P\}, \{0, 1\}, R, P)$, tendo R as 3 regras:
$P \to 0P1P \mid 1P0P \mid \lambda$

GLC para expressões aritméticas

$(\{E, T, F\}, \{\mathtt{t}, +, *, (,)\}, R, E)$, em que R consta de:

$$E \to E{+}T \mid T$$
$$T \to T{*}F \mid F$$
$$F \to (E) \mid \mathtt{t}$$

O que é linguagem livre do contexto

Linguagem livre do contexto

Uma linguagem é dita ser uma linguagem livre do contexto se existe uma gramática livre do contexto que a gera.

O conceito de árvore de derivação

Árvore de derivação

Seja $G = (V, \Sigma, R, P)$. Def. recursiva de árvore de derivação (AD):

a) uma árvore com apenas o vértice de rótulo P é uma AD;

b) se $X \in V$ é rótulo de uma folha f de uma AD, então:

i. se $X \to \lambda \in R$, então a árvore obtida acrescentando-se mais um vértice v com rótulo λ e uma aresta $\{f, v\}$ é uma AD;

ii. se $X \to x_1 x_2 \ldots x_n \in R$, onde $x_1, x_2, \ldots, x_n \in V \cup \Sigma$, então a árvore obtida acrescentando-se mais n vértices v_1, v_2, ..., v_n com rótulos x_1, x_2, ..., x_n, *nessa ordem*, e n arestas $\{f, v_1\}$, $\{f, v_2\}$, ..., $\{f, v_n\}$, é uma AD.

Exemplo de construção de uma AD para t*(t+t)

$$E \to E{+}T \mid T$$
$$T \to T{*}F \mid F$$
$$F \to (E) \mid \mathtt{t}$$

A derivação

$$E \Rightarrow T \qquad (\text{regra } E \to T)$$

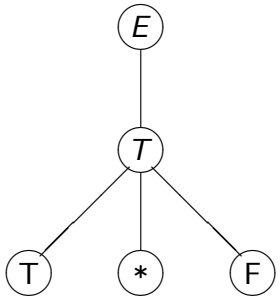
leva à AD:

Exemplo de construção de uma AD (cont.)

A derivação evolue para:

$$\begin{aligned} E &\Rightarrow T && \text{(regra } E \to T\text{)} \\ &\Rightarrow T*F && \text{(regra } T \to T*F\text{)} \end{aligned}$$

e a AD correspondente para:



Exemplo de construção de uma AD (cont.)

Tem-se duas opções para continuar a derivação:

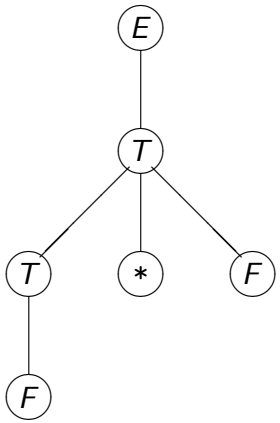
$$\begin{aligned} E &\Rightarrow T && \text{(regra } E \to T\text{)} \\ &\Rightarrow T*F && \text{(regra } T \to T*F\text{)} \\ &\Rightarrow F*F && \text{(regra } T \to F\text{)} \end{aligned}$$

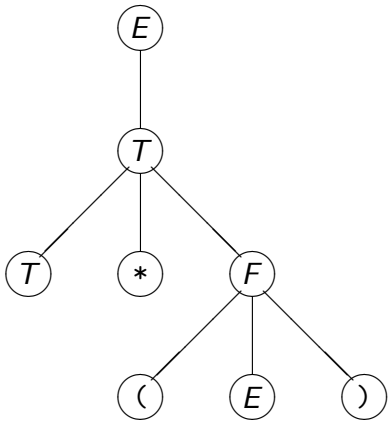
ou então:

$$\begin{aligned} E &\Rightarrow T && \text{(regra } E \to T\text{)} \\ &\Rightarrow T*F && \text{(regra } T \to T*F\text{)} \\ &\Rightarrow T*(E) && \text{(regra } F \to (E)\text{).} \end{aligned}$$

Exemplo de construção de uma AD (cont.)

As duas opções:





Exemplo de construção de uma AD/conclusão

$\Longrightarrow$ Após uma derivação de 11 passos tem-se uma AD para $\mathtt{t} * (\mathtt{t} + \mathtt{t})$.

Observações:

- Número de passos da derivação: número de vértices internos de X.
- A estrutura da AD é normalmente utilizada para associar significado.
- Mais de uma AD para $w \Rightarrow$ mais de um significado para w.

Ambiguidade

Gramática ambígua

Uma GLC é denominada ambígua quando existe mais de uma AD para alguma sentença que ela gera.

Exemplo:

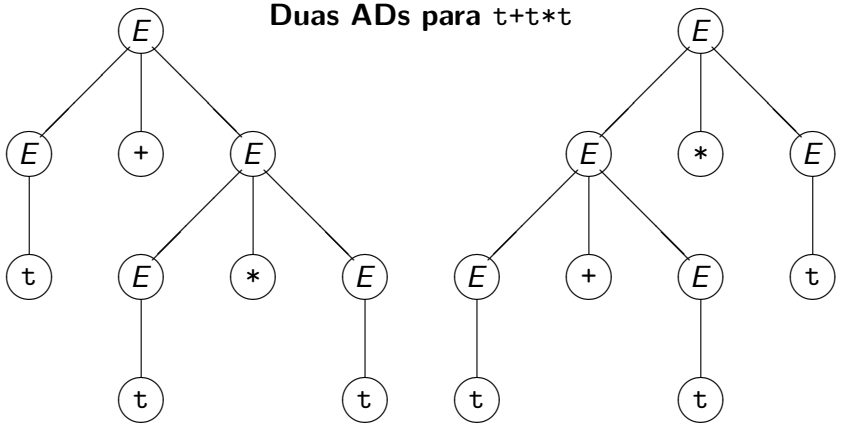
GLC ambígua para expressões aritméticas:

$$E \rightarrow E + E \mid E * E \mid (E) \mid \mathtt{t}$$

Demonstrando ambiguidade

$$E \rightarrow E + E \mid E * E \mid (E) \mid \mathtt{t}$$

Duas ADs para `t+t*t`



Derivações mais à esquerda e mais à direita

Derivação mais à esquerda

Uma derivação é dita mais à esquerda (DME) se em cada passo é expandida a variável mais à esquerda.

Derivação mais à direita

Uma derivação é dita mais à direita (DMD) se em cada passo é expandida a variável mais à direita.

$\Longrightarrow$ Existe uma única DME e uma única DMD correspondentes a uma AD e vice-versa.

Ambiguidade e DME e DMD

Como existe uma única DME e uma única DMD correspondentes a uma AD e vice-versa:

- uma GLC é ambígua se, e somente se, existe mais de uma DME para alguma sentença que ela gere;
- uma GLC é ambígua se, e somente se, existe mais de uma DMD para alguma sentença que ela gere.

Linguagens inerentemente ambíguas

Linguagem inerentemente ambígua: LLC para a qual existem apenas gramáticas ambíguas.

Exemplo:
$\{a^m b^n c^k \mid m = n \text{ ou } n = k\}$.

- A detecção e remoção de ambiguidade em GLCs é muito importante.
- O problema de determinar se uma GLC é ambígua é **indecidível**.

Variáveis inúteis

Variável útil

Seja uma GLC $G = (V, \Sigma, R, P)$. Uma variável $X \in V$ é dita ser útil se, e somente se, existem $u, v \in (V \cup \Sigma)^*$ e $w \in \Sigma^*$ tais que:

$$P \stackrel{*}{\Rightarrow} uXv \stackrel{*}{\Rightarrow} w.$$

Exemplo:

Seja a GLC:

$P \to AB \mid a$

$B \to b$

$C \to c$

Que variáveis são inúteis?

Exemplo de variáveis inúteis

$P \to AB \mid a$

$B \to b$

$C \to c$

- C é inútil: não existem u e v tais que $P \stackrel{*}{\Rightarrow} uCv$;
- A é inútil: não existe $w \in \Sigma^*$ tal que $A \stackrel{*}{\Rightarrow} w$;
- B é inútil: $P \stackrel{*}{\Rightarrow} uBv$ apenas para $u = A$ e $v = \lambda$, e não existe $w \in \Sigma^*$ tal que $AB \stackrel{*}{\Rightarrow} w$.

GLC equivalente sem símbolos inúteis:

$P \to a.$

Eliminação de variáveis inúteis

Seja $G = (V, \Sigma, R, P)$ tal que $L(G) \neq \emptyset$.
Construção de uma GLC G'' equivalente a G, sem variáveis inúteis:

a) Obtenha $G' = (V', \Sigma, R', P)$, em que:
- $V' = \{X \in V \mid X \stackrel{*}{\Rightarrow}_G w \text{ para algum } w \in \Sigma^*\}$, e
- $R' = \{r \in R \mid r \text{ não contém símbolo de } V - V'\}$.

b) Obtenha $G'' = (V'', \Sigma, R'', P)$, em que:
- $V'' = \{X \in V' \mid P \stackrel{*}{\Rightarrow}_{G'} uXv \text{ para algum } u, v \in (V' \cup \Sigma)^*\}$, e
- $R'' = \{r \in R' \mid r \text{ não contém símbolo de } V' - V''\}$.

Determinando variáveis que produzem sentenças

Algoritmo que determina $\{X \in V \mid X \stackrel{*}{\Rightarrow} w \text{ para algum } w \in \Sigma^*\}$:

Entrada: uma GLC $G = (V, \Sigma, R, P)$.
Saída: $\mathcal{I}_1 = \{X \in V \mid X \stackrel{*}{\Rightarrow} w \text{ para algum } w \in \Sigma^*\}$.
$\mathcal{I}_1 \leftarrow \emptyset$;
repita
  $\mathcal{N} \leftarrow \{X \notin \mathcal{I}_1 \mid X \to z \in R \text{ e } z \in (\mathcal{I}_1 \cup \Sigma)^*\}$;
  $\mathcal{I}_1 \leftarrow \mathcal{I}_1 \cup \mathcal{N}$
até $\mathcal{N} = \emptyset$;
retorne $\mathcal{I}_1$.

Determinando variáveis alcançáveis a partir de P

Algoritmo que determina
$\{X \in V \mid P \stackrel{*}{\Rightarrow} uXv \text{ para algum } u, v \in (V \cup \Sigma)^*\}$:

Entrada: uma GLC $G = (V, \Sigma, R, P)$.
Saída: $\mathcal{I}_2 = \{X \in V \mid P \stackrel{*}{\Rightarrow} uXv \text{ para algum } u, v \in (V \cup \Sigma)^*\}$.
$\mathcal{I}_2 \leftarrow \emptyset$; $\mathcal{N} \leftarrow \{P\}$;
repita
  $\mathcal{I}_2 \leftarrow \mathcal{I}_2 \cup \mathcal{N}$;
  $\mathcal{N} \leftarrow \{Y \notin \mathcal{I}_2 \mid X \to uYv \text{ para algum } X \in \mathcal{N} \text{ e } u, v \in (V \cup \Sigma)^*\}$
até $\mathcal{N} = \emptyset$;
retorne $\mathcal{I}_2$.

Exemplo/eliminação de variáveis inúteis

$$
\begin{aligned}
A &\to ABC \mid AEF \mid BD \\
B &\to B0 \mid 0 \\
C &\to 0C \mid EB \\
D &\to 1D \mid 1 \\
E &\to BE \\
F &\to 1F1 \mid 1
\end{aligned}
$$

$V' = \{B, D, F, A\} \Rightarrow$
$$
\begin{aligned}
A &\to BD \\
B &\to B0 \mid 0 \\
D &\to 1D \mid 1 \\
F &\to 1F1 \mid 1
\end{aligned}
$$

$V'' = \{A, B, D\} \Rightarrow$
$$
\begin{aligned}
A &\to BD \\
B &\to B0 \mid 0 \\
D &\to 1D \mid 1
\end{aligned}
$$

Eliminação de uma regra

Seja uma GLC $G = (V, \Sigma, R, P)$. Seja $X \to w \in R$, $\underline{X \neq P}$.
Seja a GLC $G' = (V, \Sigma, R', P)$ em que R' é obtido assim:

1. para cada regra de R em que X não ocorre do lado direito, exceto $X \to w$, coloque-a em R';
2. para cada regra de R da forma $Y \to x_1 X x_2 X \ldots X x_{n+1}$, com pelo menos uma ocorrência de X do lado direito, com $n \geq 1$ e $x_i \in [(V - \{X\}) \cup \Sigma]^*$, coloque em R' todas as regras da forma $Y \to x_1 \gamma_1 x_2 \gamma_2 \ldots \gamma_n x_{n+1}$, sendo que cada γ_j pode ser X ou w, com exceção da regra $X \to w$.

G' é equivalente a G.

Exemplo/eliminação de uma regra

GLC G:

$P \to ABA$

$A \to \mathtt{a}A \mid \mathtt{a}$

$B \to \mathtt{b}B\mathtt{c} \mid \lambda$

GLC G' obtida eliminando-se a regra $A \to \mathtt{a}$:

$P \to ABA \mid AB\mathtt{a} \mid \mathtt{a}BA \mid \mathtt{a}B\mathtt{a}$

$A \to \mathtt{a}A \mid \mathtt{aa}$

$B \to \mathtt{b}B\mathtt{c} \mid \lambda$

Como são as derivações de aa em G e em G'?

O conceito de variáveis anuláveis

Variável anulável

Uma variável X é anulável em uma GLC G se, e somente se, $X \stackrel{*}{\Rightarrow}_G \lambda$.

Algoritmo que determina o conjunto das variáveis anuláveis de uma GLC:

Entrada: uma GLC $G = (V, \Sigma, R, P)$;
Saída: $\{X \in V \mid X \stackrel{*}{\Rightarrow} \lambda\}$.
$\mathcal{A} \leftarrow \emptyset$;
repita
  $\mathcal{N} \leftarrow \{Y \notin \mathcal{A} \mid Y \to z \in R \text{ e } z \in \mathcal{A}^*\}$;
  $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{N}$
até $\mathcal{N} = \emptyset$;
retorne $\mathcal{A}$.

Eliminação de regras λ

Seja $G = (V, \Sigma, R, P)$. Seja $G' = (V, \Sigma, R', P)$ em que R' é obtido assim:

1. para cada regra de R cujo lado direito não contém variável anulável, exceto regra λ, coloque-a em R';
2. para cada regra de R da forma $Y \to x_1X_1x_2X_2 \ldots X_nx_{n+1}$, sendo cada X_i uma variável anulável, com $n \geq 1$ e cada x_i sem variáveis anuláveis, coloque em R' todas as regras da forma $Y \to x_1\gamma_1x_2\gamma_2 \ldots \gamma_nx_{n+1}$, em que cada γ_j pode ser X_j ou λ, com exceção de regra λ;
3. se P for anulável, coloque $P \to \lambda$ em R'.

$L(G') = L(G)$ e sua única regra λ, se houver, é $P \to \lambda$.

Exemplo/eliminação de regras λ

Seja G:

$P \to APB \mid C$

$A \to A\mathtt{aa}A \mid \lambda$

$B \to BB\mathtt{b} \mid C$

$C \to \mathtt{c}C \mid \lambda$

Variáveis anuláveis de G: $\{A, C, P, B\}$. G':

$P \to APB \mid AP \mid AB \mid PB \mid A \mid B \mid C \mid \lambda$

$A \to A\mathtt{aa}A \mid \mathtt{aa}A \mid A\mathtt{aa} \mid \mathtt{aa}$

$B \to BB\mathtt{b} \mid B\mathtt{b} \mid \mathtt{b} \mid C$

$C \to \mathtt{c}C \mid \mathtt{c}$

O conceito de variáveis encadeadas

Variável encadeada

Seja uma gramática $G = (V, \Sigma, R, P)$. Diz-se que uma variável $Z \in V$ é encadeada a uma variável $X \in V$ se $Z = X$ ou se existe uma seqüência de regras $X \to Y_1$, $Y_1 \to Y_2$, ..., $Y_n \to Z$ em R, $n \geq 0$; no caso em que $n = 0$, tem-se apenas a regra $X \to Z$. Ao conjunto de todas as variáveis encadeadas a X é dado o nome de $enc(X)$.

Algoritmo para determinar variáveis encadeadas

Algoritmo que determina $enc(X)$:

Entrada: (1) uma GLC $G = (V, \Sigma, R, P)$, e
(2) uma variável $X \in V$.
Saída: $enc(X)$.
$\mathcal{U} \leftarrow \emptyset$; $\mathcal{N} \leftarrow \{X\}$;
repita
$\mathcal{U} \leftarrow \mathcal{U} \cup \mathcal{N}$;
$\mathcal{N} \leftarrow \{Y \notin \mathcal{U} \mid Z \to Y \in R \text{ para algum } Z \in \mathcal{N}\}$
até $\mathcal{N} = \emptyset$
retorne $\mathcal{U}$.

Eliminando regras unitárias

Uma GLC equivalente a $G = (V, \Sigma, R, P)$, sem regras unitárias, é $G' = (V, \Sigma, R', P)$, em que

$R' = \{X \to w \mid \text{ existe } Y \in enc(X) \text{ tal que } Y \to w \in R \text{ e } w \notin V\}$.

Exemplo/eliminação de regras unitárias

GLC para expressões aritméticas:

$E \to E{+}T \mid T$
$T \to T{*}F \mid F$
$F \to (E) \mid \mathtt{t}$

Os conjuntos $enc(X)$ para cada variável X são:

- $enc(E) = \{E, T, F\}$;
- $enc(T) = \{T, F\}$;
- $enc(F) = \{F\}$.

GLC equivalente, sem regras unitárias:

$E \to E{+}T \mid T{*}F \mid (E) \mid \mathtt{t}$
$T \to T{*}F \mid (E) \mid \mathtt{t}$
$F \to (E) \mid \mathtt{t}$

Interação entre os métodos de eliminação

a) Ao se eliminar regras λ podem aparecer regras unitárias. Exemplo: GLC com as regras $A \to BC$ e $B \to \lambda$.

b) Ao se eliminar regras unitárias podem aparecer regras λ. Exemplo: GLC com $P \to \lambda$, sendo P é o símbolo de partida, e a regra $A \to P$.

c) Ao se eliminar regras λ podem aparecer variáveis inúteis. Exemplo: o do item (a), caso $B \to \lambda$ seja a única regra B.

d) Ao se eliminar regras unitárias podem aparecer variáveis inúteis. Exemplo: GLC que contém $A \to B$ e B não aparece do lado direito de nenhuma outra regra (B torna-se inútil).

Ao se eliminar variáveis inúteis, não podem aparecer novas regras.

Garantido consistência das eliminações

A seguinte sequência de eliminações para $G = (V, \Sigma, R, P)$:

1. adicionar a regra $P' \to P$, se P for recursivo;
2. eliminar regras λ;
3. eliminar regras unitárias;
4. eliminar símbolos inúteis.

produz uma GLC equivalente cujas regras são das formas:

- $P \to \lambda$ se $\lambda \in L(G)$;
- $X \to a$ para $a \in \Sigma$;
- $X \to w$ para $|w| \geq 2$.

Gramática na forma normal de Chomsky

Forma normal de Chomsky

Uma GLC $G = (V, \Sigma, R, P)$ está na forma normal de Chomsky (FNC) se cada uma de suas regras tem uma das formas:

- $P \to \lambda$ se $\lambda \in L(G)$;
- $X \to YZ$ para $Y, Z \in V$;
- $X \to a$ para $a \in \Sigma$.

Transformação para a forma normal de Chomsky

Seja uma GLC G. Obter uma GLC equivalente a G tal que:

- $P \to \lambda$ se $\lambda \in L(G)$;
- $X \to a$ para $a \in \Sigma$;
- $X \to w$ para $|w| \geq 2$.

Mais dois passos:

1. Modificar cada regra $X \to w$, $|w| \geq 2$, de forma que ela fique contendo apenas variáveis.
2. Substituir cada regra $X \to Y_1 Y_2 \ldots Y_n$, $n \geq 3$, em que cada Y_i é uma variável, pelo conjunto das regras: $X \to Y_1 Z_1$, $Z_1 \to Y_2 Z_2$, $\ldots$, $Z_{n-2} \to Y_{n-1} Y_n$, em que Z_1, Z_2, $\ldots$, Z_{n-2} são variáveis *novas*.

Exemplo/obtenção de gramática na FNC

Seja a GLC G:

$L \rightarrow (S)$

$S \rightarrow SE \mid \lambda$

$E \rightarrow \mathtt{a} \mid L$

Após a eliminação de regras λ:

$L \rightarrow (S) \mid ()$

$S \rightarrow SE \mid E$

$E \rightarrow \mathtt{a} \mid L$

Como $enc(L) = \{L\}$, $enc(S) = \{S, E, L\}$ e $enc(E) = \{E, L\}$:

$L \rightarrow (S) \mid ()$

$S \rightarrow SE \mid \mathtt{a} \mid (S) \mid ()$

$E \rightarrow \mathtt{a} \mid (S) \mid ()$

Exemplo/obtenção de gramática na FNC(cont.)

$L \rightarrow (S) \mid ()$

$S \rightarrow SE \mid \mathtt{a} \mid (S) \mid ()$

$E \rightarrow \mathtt{a} \mid (S) \mid ()$

Finalmente:

$L \rightarrow AX \mid AB$

$S \rightarrow SE \mid \mathtt{a} \mid AX \mid AB$

$E \rightarrow \mathtt{a} \mid AX \mid AB$

$X \rightarrow SB$

$A \rightarrow ($

$B \rightarrow)$

Eliminação de regras recursivas à esquerda

Sejam as regras, a seguir, **todas** as regras X de uma GLC G:

$X \rightarrow Xy_1 \mid Xy_2 \mid \ldots \mid Xy_n \mid w_1 \mid w_2 \mid \ldots \mid w_k$

em que nenhum w_i começa com X.

Utilizando recursão à direita, em vez de recursão à esquerda:

$X \rightarrow w_1Z \mid w_2Z \mid \ldots \mid w_kZ$

$Z \rightarrow y_1Z \mid y_2Z \mid \ldots \mid y_nZ \mid \lambda$

em que Z é uma variável *nova*. Eliminado-se a regra λ:

$X \rightarrow w_1Z \mid w_2Z \mid \ldots \mid w_kZ \mid w_1 \mid w_2 \mid \ldots \mid w_k$

$Z \rightarrow y_1Z \mid y_2Z \mid \ldots \mid y_nZ \mid y_1 \mid y_2 \mid \ldots \mid y_n$

Exemplo/eliminação de regras recursivas à esquerda

G:

$E \rightarrow E{+}E \mid E{*}E \mid (E) \mid \mathtt{t}$

Retirando-se recursão à esquerda:

$E \rightarrow (E)Z \mid \mathtt{t}Z$

$Z \rightarrow {+}EZ \mid {*}EZ \mid \lambda$

Exemplo/eliminação de variável em regra

Seja $G = (V, \Sigma, R, P)$ tal que $X \to uYv \in R$, $Y \in V$ e $Y \neq X$.

Sejam $Y \to w_1 | w_2 | \ldots | w_n$ *todas* as regras Y em R.

R pode ser substituída por

$$(R - \{X \to uYv\}) \cup \{X \to uw_1v \mid uw_2v \mid \ldots \mid uw_nv\}.$$

Gramática na forma normal de Greibach

Forma normal de Greibach

Uma GLC $G = (V, \Sigma, R, P)$ está na forma normal de Greibach (FNG) se cada uma de suas regras tem uma das formas:

- $P \to \lambda$ se $\lambda \in L(G)$;
- $X \to ay$ para $a \in \Sigma$ e $y \in V^*$.

Uma forma sentencial de uma gramática na FNG, com exceção de λ, é da forma xy, em que $x \in \Sigma^+$ e $y \in V^*$.

Qual o tamanho de uma derivação de uma palavra?

Transformação para a forma normal de Greibach

Seja uma GLC G. Obter uma GLC equivalente a G tal que:

- $P \to \lambda$ se $\lambda \in L(G)$;
- $X \to a$ para $a \in \Sigma$;
- $X \to w$ para $|w| \geq 2$.

Em seguida:

Para cada $X \to w$ para $|w| \geq 2$, substituir por variáveis todos os terminais de w, a partir de seu segundo símbolo. Com isso, obtêm-se regras da forma $X \to Yy$, sendo que $Y \in V \cup \Sigma$ e $y \in V^+$.

Transformação para a FNG (cont.)

- Numerar sequencialmente as variáveis ($\#P = 1$).
- Para cada $A \in V$, começando com P, na ordem dada pela numeração, fazer enquanto possível:
  1. Se existe $A \to By$, para $|y| \geq 1$, tal que $\#A > \#B$, substituir B.
  2. Se existe $A \to Ay$, para $|y| \geq 1$, eliminar a recursão à esquerda.
- Seja B a variável de maior número em que há regra da forma $B \to Cy$, com $C \in V$. Substituir C. Repetir até ser atingida a variável P.
- Para regras da forma $Z \to Ay$, em que Z é variável nova criada eliminando-se recursão à esquerda, substituir A.

Exemplo/transformação para a FNG

G:

$A \to CB$
$B \to BBD \mid \mathtt{b}$
$C \to BBC \mid D\mathtt{c}$
$D \to AD \mid \mathtt{d}$

Elimina-se $C \to D\mathtt{c}$ e introduz-se $C \to DE$ e $E \to \mathtt{c}$.

Numeração: $\#A = 1$, $\#B = 2$, $\#C = 3$, $\#D = 4$ e $\#E = 5$.

$A \to CB$
$*\ B \to \mathtt{b} \mid \mathtt{b}Z_1$
$C \to BBC \mid DE$
$D \to AD \mid \mathtt{d}$
$E \to \mathtt{c}$
$*\ Z_1 \to BD \mid BDZ_1$

Exemplo/transformação para a FNG (cont.)

Substituindo-se a regra $C \to BBC$ ($\#C > \#B$):

$A \to CB$
$B \to \mathtt{b} \mid \mathtt{b}Z_1$
$*\ C \to \mathtt{b}BC \mid \mathtt{b}Z_1BC$
$C \to DE$
$D \to AD \mid \mathtt{d}$
$E \to \mathtt{c}$
$Z_1 \to BD \mid BDZ_1$

Substituindo-se a regra $D \to AD$ ($\#D > \#A$):

$A \to CB$
$B \to \mathtt{b} \mid \mathtt{b}Z_1$
$C \to \mathtt{b}BC \mid \mathtt{b}Z_1BC \mid DE$
$*\ D \to CBD$
$D \to \mathtt{d}$
$E \to \mathtt{c}$
$Z_1 \to BD \mid BDZ_1$

Exemplo/transformação para a FNG (cont.)

Substituindo-se a regra $D \to CBD$ ($\#D > \#C$):

$A \to CB$
$B \to \mathtt{b} \mid \mathtt{b}Z_1$
$C \to \mathtt{b}BC \mid \mathtt{b}Z_1BC \mid DE$
$*\ D \to \mathtt{b}BCBD \mid \mathtt{b}Z_1BCBD \mid DEBD$
$D \to \mathtt{d}$
$E \to \mathtt{c}$
$Z_1 \to BD \mid BDZ_1$

Eliminando-se a recursão à esquerda para as regras D:

$A \to CB$
$B \to \mathtt{b} \mid \mathtt{b}Z_1$
$C \to \mathtt{b}BC \mid \mathtt{b}Z_1BC \mid DE$
$*\ D \to \mathtt{b}BCBD \mid \mathtt{b}Z_1BCBD \mid \mathtt{d} \mid \mathtt{b}BCBDZ_2 \mid \mathtt{b}Z_1BCBDZ_2 \mid \mathtt{d}Z_2$
$E \to \mathtt{c}$
$Z_1 \to BD \mid BDZ_1$
$*\ Z_2 \to EBD \mid EBDZ_2$

Exemplo/transformação para a FNG (cont.)

Substituindo-se $C \to DE$:

$A \to CB$
$B \to \mathtt{b} \mid \mathtt{b}Z_1$
$C \to \mathtt{b}BC \mid \mathtt{b}Z_1BC$
$*\ C \to \mathtt{b}BCBDE \mid \mathtt{b}Z_1BCBDE \mid \mathtt{d}E \mid \mathtt{b}BCBDZ_2E$
$\quad \mid \mathtt{b}Z_1BCBDZ_2E \mid \mathtt{d}Z_2E$
$D \to \mathtt{b}BCBD \mid \mathtt{b}Z_1BCBD \mid \mathtt{d} \mid \mathtt{b}BCBDZ_2 \mid \mathtt{b}Z_1BCBDZ_2 \mid \mathtt{d}Z_2$
$E \to \mathtt{c}$
$Z_1 \to BD \mid BDZ_1$
$Z_2 \to EBD \mid EBDZ_2$

Exemplo/transformação para a FNG (cont.)

Substituindo-se $A \to CB$:

* $A \to \mathtt{b}BCB \mid \mathtt{b}Z_1BCB \mid \mathtt{b}BCBDEB \mid \mathtt{b}Z_1BCBDEB \mid \mathtt{d}EB$
  $\mid \mathtt{b}BCBDZ_2EB \mid \mathtt{b}Z_1BCBDZ_2EB \mid \mathtt{d}Z_2EB$
  $B \to \mathtt{b} \mid \mathtt{b}Z_1$
  $C \to \mathtt{b}BC \mid \mathtt{b}Z_1BC \mid \mathtt{b}BCBDE \mid \mathtt{b}Z_1BCBDE$
  $\mid \mathtt{d}E \mid \mathtt{b}BCBDZ_2E \mid \mathtt{b}Z_1BCBDZ_2E \mid \mathtt{d}Z_2E$
  $D \to \mathtt{b}BCBD \mid \mathtt{b}Z_1BCBD \mid \mathtt{d} \mid \mathtt{b}BCBDZ_2 \mid \mathtt{b}Z_1BCBDZ_2 \mid \mathtt{d}Z_2$
  $E \to \mathtt{c}$
  $Z_1 \to BD \mid BDZ_1$
  $Z_2 \to EBD \mid EBDZ_2$

Exemplo/transformação para a FNG (cont.)

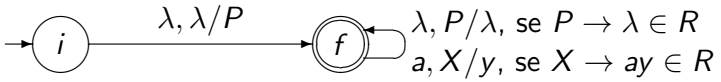
Finalmente, substitui-se as regras introduzidas por eliminação de recursão à esquerda:

$A \to \mathtt{b}BCB \mid \mathtt{b}Z_1BCB \mid \mathtt{b}BCBDEB \mid \mathtt{b}Z_1BCBDEB \mid \mathtt{d}EB$
$\mid \mathtt{b}BCBDZ_2EB \mid \mathtt{b}Z_1BCBDZ_2EB \mid \mathtt{d}Z_2EB$
$B \to \mathtt{b} \mid \mathtt{b}Z_1$
$C \to \mathtt{b}BC \mid \mathtt{b}Z_1BC \mid \mathtt{b}BCBDE \mid \mathtt{b}Z_1BCBDE$
$\mid \mathtt{d}E \mid \mathtt{b}BCBDZ_2E \mid \mathtt{b}Z_1BCBDZ_2E \mid \mathtt{d}Z_2E$
$D \to \mathtt{b}BCBD \mid \mathtt{b}Z_1BCBD \mid \mathtt{d} \mid \mathtt{b}BCBDZ_2 \mid \mathtt{b}Z_1BCBDZ_2 \mid \mathtt{d}Z_2$
$E \to \mathtt{c}$
* $Z_1 \to \mathtt{b}D \mid \mathtt{b}Z_1D \mid \mathtt{b}DZ_1 \mid \mathtt{b}Z_1DZ_1$
* $Z_2 \to \mathtt{c}BD \mid \mathtt{c}BDZ_2$

Obtenção de AP a partir de GLC

Seja $G' = (V, \Sigma, R, P)$ uma GLC na FNG equivalente a G.

APN que aceita $L(G')$:



Obtenção de AP a partir de GLC/Exemplo

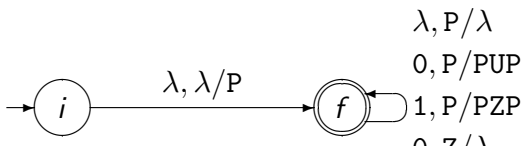
Seja $G = (\{P\}, \{0, 1\}, R, P)$, em que R consta das regras:

$P \to 0P1P \mid 1P0P \mid \lambda$

Uma GLC equivalente na FNG:

$P \to 0PUP \mid 1PZP \mid \lambda$
$Z \to 0$
$U \to 1$

Um AP que reconhece $L(G)$:



Obtenção de GLC a partir de AP

A idéia:
Seja um APN $M = (E, \Sigma, \Gamma, \delta, I, F)$. Dados $e, e' \in E$ e $X \in \Gamma \cup \{\lambda\}$:

$$C(e, X, e') = \{w \in \Sigma^* \mid [e, w, X] \overset{*}{\vdash} [e', \lambda, \lambda]\}.$$

Então:

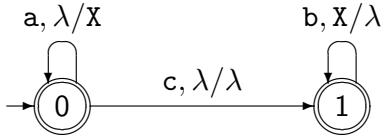
$$L(M) = \bigcup_{(i,f) \in I \times F} C(i, \lambda, f).$$

Obtenção da GLC:

1. Criar regras $P \to [i, \lambda, f]$, para cada $i \in I$ e $f \in F$.
2. Criar regras para variáveis $[e, X, e']$, de tal forma que de $[e, X, e']$ se gere o conjunto $C(e, X, e')$.

Obtenção de GLC a partir de AP/Exemplo

Seja o APD:



Primeiras regras:

- $P \to [0, \lambda, 0] \mid [0, \lambda, 1]$

Regras $[0, \lambda, 0]$:

- $[0, \lambda, 0] \to \lambda$
- $[0, \lambda, 0] \to \mathtt{a}[0, \mathtt{X}, 0]$
- $[0, \lambda, 0] \to \mathtt{c}[1, \lambda, 0]$

As variáveis $[0, \mathtt{X}, 0]$ e $[1, \lambda, 0]$ são inúteis!

Como são as outras regras?