MAC 414
Autômatos, Computabilidade e Complexidade
aula 9 — 14/10/2020

Linguagens livres de contexto — um apanhado geral —

Linguagens livres de contexto — um apanhado geral —

Importância: praticamente todas as linguagens de programação nascem LC.

Linguagens livres de contexto — um apanhado geral —

Importância: praticamente todas as linguagens de programação nascem LC.

Vale para linguagens declarativas, como html, json,...

Linguagens livres de contexto — um apanhado geral —

Importância: praticamente todas as linguagens de programação nascem LC.

Vale para linguagens declarativas, como html, json,...

Razões:

- Formas convenientes de especificar

Linguagens livres de contexto — um apanhado geral —

Importância: praticamente todas as linguagens de programação nascem LC.

Vale para linguagens declarativas, como html, json,...

Razões:

- Formas convenientes de especificar
- Se especificada com cuidado, fácil de analisar

Linguagens livres de contexto — um apanhado geral —

Importância: praticamente todas as linguagens de programação nascem LC.

Vale para linguagens declarativas, como html, json,...

Razões:

- Formas convenientes de especificar
- Se especificada com cuidado, fácil de analisar
- Possível associar comportamento a várias fases da análise (p.e., geração de código)

Gramática

Uma gramática LC $G = (V, \Sigma, R, S)$ (Chomsky, 1956) consiste de:

- Um alfabeto V,
- Um alfabeto $\Sigma \subset V$, de terminais,
- Um subconjunto *finito* R de $(V \backslash \Sigma) \times \Sigma^*$, de regras,
- O símbolo inicial $S \in V \backslash \Sigma$.

Gramática

Uma gramática LC $G = (V, \Sigma, R, S)$ (Chomsky, 1956) consiste de:

- Um alfabeto V,
- Um alfabeto $\Sigma \subset V$, de terminais,
- Um subconjunto *finito* R de $(V \backslash \Sigma) \times \Sigma^*$, de regras,
- O símbolo inicial $S \in V \backslash \Sigma$.

Os membros de $V \backslash \Sigma$ são os não terminais.

Gramática

Uma gramática LC $G = (V, \Sigma, R, S)$ (Chomsky, 1956) consiste de:

- Um alfabeto V,
- Um alfabeto $\Sigma \subset V$, de terminais,
- Um subconjunto *finito* R de $(V \setminus \Sigma) \times \Sigma^*$, de regras,
- O símbolo inicial $S \in V \setminus \Sigma$.

Os membros de $V \setminus \Sigma$ são os não terminais.

Escrevemos $A \to_G v$ se $(A, v) \in R$.

Derivações

Derivações

Vamos definir a relação $\underset{G}{\Rightarrow}$ sobre V^* por $u \underset{G}{\Rightarrow} v$ se existem fatorações $u = xAy$, $v = xwy$, onde $A \underset{G}{\rightarrow} w$.

Derivações

Vamos definir a relação $\underset{G}{\Rightarrow}$ sobre V^* por $u \underset{G}{\Rightarrow} v$ se existem fatorações $u = xAy$, $v = xwy$, onde $A \underset{G}{\rightarrow} w$. (derivação em 1 passo).

Derivações

Vamos definir a relação $\underset{G}{\Rightarrow}$ sobre V^* por $u \underset{G}{\Rightarrow} v$ se existem fatorações $u = xAy$, $v = xwy$, onde $A \underset{G}{\rightarrow} w$. (derivação em 1 passo).

A partir daí definimos $\underset{G}{\overset{*}{\Rightarrow}}$ como o fecho reflexivo-transitivo de $\underset{G}{\Rightarrow}$,

Derivações

Vamos definir a relação $\underset{G}{\Rightarrow}$ sobre V^* por $u \underset{G}{\Rightarrow} v$ se existem fatorações $u = xAy$, $v = xwy$, onde $A \underset{G}{\rightarrow} w$. (derivação em 1 passo).

A partir daí definimos $\underset{G}{\Rightarrow}^*$ como o fecho reflexivo-transitivo de $\underset{G}{\Rightarrow}$, isto é considere o grafo dirigido que tem V^* como vértices e arestas dirigidas dadas por $\underset{G}{\Rightarrow}$. Então $u \underset{G}{\Rightarrow}^* v$ significa que existe passeio de u a v.

Derivações

Vamos definir a relação $\Rightarrow_G$ sobre V^* por $u \Rightarrow_G v$ se existem fatorações $u = xAy$, $v = xwy$, onde $A \rightarrow_G w$. (derivação em 1 passo).

A partir daí definimos $\Rightarrow_G^*$ como o fecho reflexivo-transitivo de $\Rightarrow_G$, isto é considere o grafo dirigido que tem V^* como vértices e arestas dirigidas dadas por $\Rightarrow_G$. Então $u \Rightarrow_G^* v$ significa que existe passeio de u a v.

A linguagem gerada por G é

$$L(G) = \{x \in \Sigma^* \mid S \Rightarrow_G^* x\}.$$

Exemplo

Exemplo

$S \rightarrow aSb \mid \lambda$

$S \rightarrow aSb, \; S \rightarrow \lambda$

Exemplo

$S \to aSb \mid \lambda$

$S \underset{G}{\Rightarrow} \lambda \in L(G)$

Exemplo

$S \to aSb \mid \lambda$

$S \underset{G}{\Rightarrow} \lambda \in L(G)$

$S \underset{G}{\Rightarrow} aSb \underset{G}{\Rightarrow} ab \in L(G)$

Exemplo

$S \rightarrow aSb \mid \lambda$

$S \underset{G}{\Rightarrow} \lambda \in L(G)$

$S \underset{G}{\Rightarrow} aSb \underset{G}{\Rightarrow} ab \in L(G)$

$S \underset{G}{\Rightarrow} aSb \underset{G}{\Rightarrow} aaSbb \underset{G}{\Rightarrow} a^2b^2 \in L(G)$

Exemplo

$S \rightarrow aSb \mid \lambda$

$S \underset{G}{\Rightarrow} \lambda \in L(G)$

$S \underset{G}{\Rightarrow} aSb \underset{G}{\Rightarrow} ab \in L(G)$

$S \underset{G}{\Rightarrow} aSb \underset{G}{\Rightarrow} aaSbb \underset{G}{\Rightarrow} a^2b^2 \in L(G)$

É fácil ver que

$$S \overset{*}{\underset{G}{\Rightarrow}} x \quad \text{sse} \quad x = a^n S b^n \text{ ou } a^n b^n$$

Exemplo

$S \rightarrow aSb \mid \lambda$

$S \underset{G}{\Rightarrow} \lambda \in L(G)$

$S \underset{G}{\Rightarrow} aSb \underset{G}{\Rightarrow} ab \in L(G)$

$S \underset{G}{\Rightarrow} aSb \underset{G}{\Rightarrow} aaSbb \underset{G}{\Rightarrow} a^2b^2 \in L(G)$

É fácil ver que

$$S \overset{*}{\underset{G}{\Rightarrow}} x \quad \text{sse} \quad x = a^n S b^n \text{ ou } a^n b^n$$

Logo: $L(G) = AnBn$

Exemplo

Exemplo

$S \rightarrow (S) \mid SS \mid \lambda$

Exemplo

$$S \to (S) \mid SS \mid \lambda$$

$$S \underset{G}{\Rightarrow} SS \underset{G}{\Rightarrow} S(S) \underset{G}{\Rightarrow} S(SSS) \underset{G}{\Rightarrow}^{*} ()(()())$$

Exemplo

$S \rightarrow (S) \mid SS \mid \lambda$

$S \underset{G}{\Rightarrow} SS \underset{G}{\Rightarrow} S(S) \underset{G}{\Rightarrow} S(SSS) \underset{G}{\Rightarrow}^{*} ()(()())$

$L(G)$= parênteses bem formados: cada prefixo tem pelo menos tantos (quantos)

Exemplo

$S \rightarrow (S) \mid SS \mid \lambda$

$S \underset{G}{\Rightarrow} SS \underset{G}{\Rightarrow} S(S) \underset{G}{\Rightarrow} S(SSS) \underset{G}{\Rightarrow}^{*} ()(()())$

$L(G)$= parênteses bem formados: cada prefixo tem pelo menos tantos (quantos)

Outra gramática:
$S \rightarrow (S)S \mid \lambda$

Exemplo

Exemplo

$S \to E \mid E{+}S,$
$E \to T \mid TE,$
$T \to F \mid T^*,$
$F \to (S) \mid A,$
$A \to a \mid b \mid \lambda$

Exemplo

$S \rightarrow E \mid E{+}S,$
$E \rightarrow T \mid TE,$
$T \rightarrow F \mid T^{*},$
$F \rightarrow (S) \mid A,$
$A \rightarrow a \mid b \mid \lambda$

$$S \underset{G}{\Rightarrow} E{+}S \underset{G}{\Rightarrow} TE{+}S \underset{G}{\Rightarrow} TTE{+}S \underset{G}{\Rightarrow}$$

Exemplo

$S \to E \mid E+S,$
$E \to T \mid TE,$
$T \to F \mid T^*,$
$F \to (S) \mid A,$
$A \to a \mid b \mid \lambda$

$$S \Rightarrow_G E+S \Rightarrow_G TE+S \Rightarrow_G TTE+S \Rightarrow_G$$
$$TTTE+S \Rightarrow_G^* TTT^*+S \Rightarrow_G^*$$

Exemplo

$$S \to E \mid E{+}S,$$
$$E \to T \mid TE,$$
$$T \to F \mid T^*,$$
$$F \to (S) \mid A,$$
$$A \to a \mid b \mid \lambda$$

$$S \Rightarrow_G E{+}S \Rightarrow_G TE{+}S \Rightarrow_G TTE{+}S \Rightarrow_G$$
$$TTTE{+}S \Rightarrow_G^* TTT^*{+}S \Rightarrow_G^*$$
$$FTT^*{+}S \Rightarrow_G^* AAT^*{+}S \Rightarrow_G^*$$

Exemplo

$S \to E \mid E{+}S,$
$E \to T \mid TE,$
$T \to F \mid T^*,$
$F \to (S) \mid A,$
$A \to a \mid b \mid \lambda$

$$S \underset{G}{\Rightarrow} E{+}S \underset{G}{\Rightarrow} TE{+}S \underset{G}{\Rightarrow} TTE{+}S \underset{G}{\Rightarrow}$$
$$TTTE{+}S \underset{G}{\Rightarrow}^* TTT^*{+}S \underset{G}{\Rightarrow}^*$$
$$FTT^*{+}S \underset{G}{\Rightarrow}^* AAT^*{+}S \underset{G}{\Rightarrow}^*$$
$$ab(S)^*{+}S \underset{G}{\Rightarrow}^*$$

Exemplo

$$S \to E \mid E{+}S,$$
$$E \to T \mid TE,$$
$$T \to F \mid T^{*},$$
$$F \to (S) \mid A,$$
$$A \to a \mid b \mid \lambda$$

$$S \underset{G}{\Rightarrow} E{+}S \underset{G}{\Rightarrow} TE{+}S \underset{G}{\Rightarrow} TTE{+}S \underset{G}{\Rightarrow}$$
$$TTTE{+}S \overset{*}{\underset{G}{\Rightarrow}} TTT^{*}{+}S \overset{*}{\underset{G}{\Rightarrow}}$$
$$FTT^{*}{+}S \overset{*}{\underset{G}{\Rightarrow}} AAT^{*}{+}S \overset{*}{\underset{G}{\Rightarrow}}$$
$$ab(S)^{*}{+}S \overset{*}{\underset{G}{\Rightarrow}}$$
$$ab(a{+}ab)^{*}{+}(aa{+}bb(a{+}b)^{*})^{*}$$

Exemplo

$S \to E \mid E{+}S,$
$E \to T \mid TE,$
$T \to F \mid T^*,$
$F \to (S) \mid A,$
$A \to a \mid b \mid \lambda$

$$S \Rightarrow_G E{+}S \Rightarrow_G TE{+}S \Rightarrow_G TTE{+}S \Rightarrow_G$$
$$TTTE{+}S \Rightarrow_G^* TTT^*{+}S \Rightarrow_G^*$$
$$FTT^*{+}S \Rightarrow_G^* AAT^*{+}S \Rightarrow_G^*$$
$$ab(S)^*{+}S \Rightarrow_G^*$$
$$ab(a{+}ab)^*{+}(aa{+}bb(a{+}b)^*)^*$$

Exemplo

$$
\begin{aligned}
S &\to E \mid E{+}S, \\
E &\to T \mid TE, \\
T &\to F \mid T^{*}, \\
F &\to (S) \mid A, \\
A &\to a \mid b \mid \lambda
\end{aligned}
$$

$$
\begin{aligned}
S \underset{G}{\Rightarrow} E{+}S \underset{G}{\Rightarrow} TE{+}S \underset{G}{\Rightarrow} TTE{+}S \underset{G}{\Rightarrow} \\
TTTE{+}S \underset{G}{\Rightarrow}^{*} TTT^{*}{+}S \underset{G}{\Rightarrow}^{*} \\
FTT^{*}{+}S \underset{G}{\Rightarrow}^{*} AAT^{*}{+}S \underset{G}{\Rightarrow}^{*} \\
ab(S)^{*}{+}S \underset{G}{\Rightarrow}^{*} \\
ab(a{+}ab)^{*}{+}(aa{+}bb(a{+}b)^{*})^{*}
\end{aligned}
$$

$L(G) =$ expressões regulares sobre $\{a, b\}$.

Extraído do Pascal

Backus-Naur Form (BNF) (1959, Algol)
Não-terminal entre <>
::= para $\rightarrow$

Extraído do Pascal

Backus-Naur Form (BNF) (1959, Algol)
Não-terminal entre <>
::= para $\rightarrow$

```
<simple expression> ::= <term> | <sign> <term>| <simple expression> <adding operator> <term>

<adding operator> ::= + | - | or

<term> ::= <factor> | <term> <multiplying operator> <factor>

<multiplying operator> ::= * | / | div | mod | and

<factor> ::= <variable> | <unsigned constant> | ( <expression> ) |
             <function designator> | <set> | not <factor>

<unsigned constant> ::= <unsigned number> | <string> | < constant identifier> < nil>
```

Gramáticas lineares

Gramáticas lineares

Regras da forma $A \to \sigma B$ e $A \to \lambda$.

Gramáticas lineares

Regras da forma $A \to \sigma B$ e $A \to \lambda$.

Fácil construir um AND em que passeios vencedores correspondem a derivações.

Gramáticas lineares

Regras da forma $A \to \sigma B$ e $A \to \lambda$.

Fácil construir um AND em que passeios vencedores correspondem a derivações.

e vice-versa: dado AND, produzir gramática.

Gramáticas lineares

Regras da forma $A \to \sigma B$ e $A \to \lambda$.

Fácil construir um AND em que passeios vencedores correspondem a derivações.

e vice-versa: dado AND, produzir gramática.

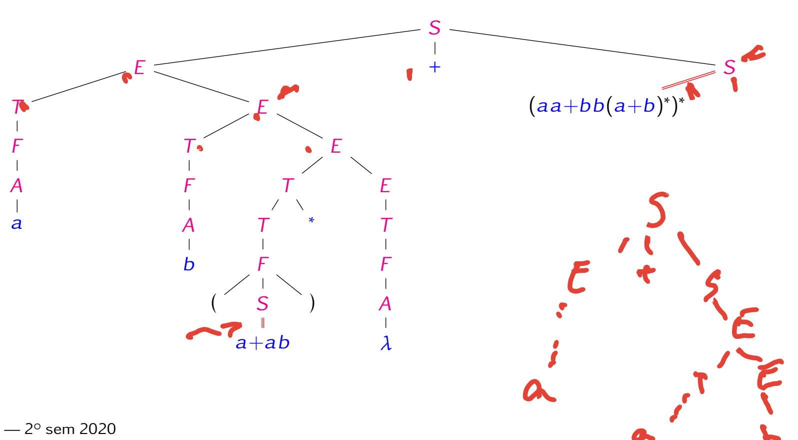
Segue que *toda linguagem regular é LC*.

Árvore de derivação

$S \Rightarrow_G E{+}S \Rightarrow_G TE{+}S \Rightarrow_G TTE{+}S \Rightarrow_G TTTE{+}S \Rightarrow_G^* TTT^*{+}S \Rightarrow_G^* FTT^*{+}S \Rightarrow_G^* AAT^*{+}S \Rightarrow_G^* ab(S)^*{+}S \Rightarrow_G^*$
$ab(a{+}ab)^*{+}(aa{+}bb(a{+}b)^*)^*$

Árvore de derivação

$S \Rightarrow_G E{+}S \Rightarrow_G TE{+}S \Rightarrow_G TTE{+}S \Rightarrow_G TTTE{+}S \Rightarrow_G^* TTT^*{+}S \Rightarrow_G^* FTT^*{+}S \Rightarrow_G^* AAT^*{+}S \Rightarrow_G^* ab(S)^*{+}S \Rightarrow_G^* ab(a{+}ab)^*{+}(aa{+}bb(a{+}b)^*)^*$



Operações com LC

Operações com LC

Prop: A família de linguagens LC é fechada por união, produto e estrela.

Operações com LC

Prop: A família de linguagens LC é fechada por união, produto e estrela.

Dem: (a menos de muitos detalhes). Considere gramáticas $G_i = (V_i, \Sigma, R_i, S_i), i = 1, 2$, $V_1 \cap V_2 = \Sigma$.

Operações com LC

Prop: A família de linguagens LC é fechada por união, produto e estrela.

Dem: (a menos de muitos detalhes). Considere gramáticas $G_i = (V_i, \Sigma, R_i, S_i), i = 1, 2,$ $V_1 \cap V_2 = \Sigma$.

Junte $G_1, G_2, S \rightarrow S_1 | S_2$, gera $L_1 \cup L_2$.

Operações com LC

Prop: A família de linguagens LC é fechada por união, produto e estrela.

Dem: (a menos de muitos detalhes). Considere gramáticas $G_i = (V_i, \Sigma, R_i, S_i), i = 1, 2,$ $V_1 \cap V_2 = \Sigma$.

Junte $G_1, G_2, S \rightarrow S_1 | S_2$, gera $L_1 \cup L_2$.

Junte $G_1, G_2, S \rightarrow S_1 S_2$, gera $L_1 L_2$.

Operações com LC

Prop: A família de linguagens LC é fechada por união, produto e estrela.

Dem: (a menos de muitos detalhes). Considere gramáticas $G_i = (V_i, \Sigma, R_i, S_i), i = 1, 2,$ $V_1 \cap V_2 = \Sigma$.

Junte $G_1, G_2, S \to S_1 | S_2$, gera $L_1 \cup L_2$.

Junte $G_1, G_2, S \to S_1 S_2$, gera $L_1 L_2$.

Junte $G_1, S \to S S_1 | \lambda$, gera L_1^*. $\square$

Linguagens não LC

Teorema

Se L é LC, então existe N tal que toda $w \in L$ de comprimento pelo menos N pode ser escrita como $w = uvxyz$, de forma que $vy \neq \lambda$ e para todo $n \geq 0$, $uv^nxy^nz \in L$.

Linguagens não LC

Teorema

Se L é LC, então existe N tal que toda $w \in L$ de comprimento pelo menos N pode ser escrita como $w = uvxyz$, de forma que $vy \neq \lambda$ e para todo $n \geq 0$, $uv^nxy^nz \in L$.

Linguagens não LC

Teorema

Se L é LC, então existe N tal que toda $w \in L$ de comprimento pelo menos N pode ser escrita como $w = uvxyz$, de forma que $vy \neq \lambda$ e para todo $n \geq 0$, $uv^n xy^n z \in L$.

Ex: $\{a^n b^n c^n \mid n \geq 0\}$ não é LC.

Linguagens não LC

Teorema

Se L é LC, então existe N tal que toda $w \in L$ de comprimento pelo menos N pode ser escrita como $w = uvxyz$, de forma que $vy \neq \lambda$ e para todo $n \geq 0$, $uv^n xy^n z \in L$.

Ex: $\{a^n b^n c^n \mid n \geq 0\}$ não é LC.

Fato: LC não é fechada por interseção nem complemento.

Linguagens não LC

Teorema

Se L é LC, então existe N tal que toda $w \in L$ de comprimento pelo menos N pode ser escrita como $w = uvxyz$, de forma que $vy \neq \lambda$ e para todo $n \geq 0$, $uv^nxy^nz \in L$.

Ex: $\{a^nb^nc^n \mid n \geq 0\}$ não é LC.

Fato: LC não é fechada por interseção nem complemento.

$\{a^nb^nc^m \mid n, m \geq 0\} \cap \{a^nb^mc^m \mid n, m \geq 0\} = \{a^nb^nc^n \mid n \geq 0\}$

Linguagens não LC

Teorema

Se L é LC, então existe N tal que toda $w \in L$ de comprimento pelo menos N pode ser escrita como $w = uvxyz$, de forma que $vy \neq \lambda$ e para todo $n \geq 0$, $uv^n xy^n z \in L$.

Ex: $\{a^n b^n c^n \mid n \geq 0\}$ não é LC.

Fato: LC não é fechada por interseção nem complemento.

$\{a^n b^n c^m \mid n, m \geq 0\} \cap \{a^n b^m c^m \mid n, m \geq 0\} = \{a^n b^n c^n \mid n \geq 0\}$

Mas, interseção de LC com regular é LC.

Como decidir pertinência?

Dada G e $x \in \Sigma^*$, $x \in L(G)$?

Como decidir pertinência?

Dada G e $x \in \Sigma^*$, $x \in L(G)$?

Tamanho de G: $|G| = \sum_{A \to v \in R}(1 + |v|) \quad + |V| + |\Sigma|$

Como decidir pertinência?

Dada G e $x \in \Sigma^*$, $x \in L(G)$?

Tamanho de G: $|G| = \sum_{A \to v \in R}(1 + |v|) \quad + |V| + |\Sigma|$

Gerar todas as derivações até obter palavras de comprimento $> |x|$ (regras $A \to \lambda$ são um problema, tratável).

Como decidir pertinência?

Dada G e $x \in \Sigma^*$, $x \in L(G)$?

Tamanho de G: $|G| = \sum_{A \to v \in R}(1 + |v|) \quad + |V| + |\Sigma|$

Gerar todas as derivações até obter palavras de comprimento $> |x|$ (regras $A \to \lambda$ são um problema, tratável). Exponencial em $|x|$

Como decidir pertinência?

Dada G e $x \in \Sigma^*$, $x \in L(G)$?

Tamanho de G: $|G| = \sum_{A \to v \in R}(1 + |v|) \quad + |V| + |\Sigma|$

Gerar todas as derivações até obter palavras de comprimento $> |x|$ (regras $A \to \lambda$ são um problema, tratável). Exponencial em $|x|$

Surpresa: existe algoritmo baseado em Programação Dinâmica que resolve o problema em tempo $\mathcal{O}(|x|^3)$. Envolve um pré-processamento de G em tempo polinomial.

$A \to BC \quad A \to \sigma$

Como decidir pertinência?

Dada G e $x \in \Sigma^*$, $x \in L(G)$?

Tamanho de G: $|G| = \sum_{A \to v \in R}(1 + |v|) \quad + |V| + |\Sigma|$

Gerar todas as derivações até obter palavras de comprimento $> |x|$ (regras $A \to \lambda$ são um problema, tratável). Exponencial em $|x|$

Surpresa: existe algoritmo baseado em Programação Dinâmica que resolve o problema em tempo $\mathcal{O}(|x|^3)$. Envolve um pré-processamento de G em tempo polinomial.

Não serve na prática.

Autômato com pilha

Autômato com pilha

Idéia:

- Controle finito, estado inicial, estados finais.

Autômato com pilha

Idéia:

- Controle finito, estado inicial, estados finais.
- Entrada: fita lida numa direção (como num AD).

Autômato com pilha

Idéia:

- Controle finito, estado inicial, estados finais.
- Entrada: fita lida numa direção (como num AD).
- Memória auxiliar: pilha. `PUSH, POP, ISEMPTY`

Autômato com pilha

Idéia:

- Controle finito, estado inicial, estados finais.
- Entrada: fita lida numa direção (como num AD).
- Memória auxiliar: pilha. `PUSH, POP, ISEMPTY`

Autômato com pilha

Idéia:

- Controle finito, estado inicial, estados finais.
- Entrada: fita lida numa direção (como num AD).
- Memória auxiliar: pilha. `PUSH, POP, ISEMPTY`

Funcionamento:

- Começa no estado inicial

Autômato com pilha

Idéia:

- Controle finito, estado inicial, estados finais.
- Entrada: fita lida numa direção (como num AD).
- Memória auxiliar: pilha. `PUSH, POP, ISEMPTY`

Funcionamento:

- Começa no estado inicial
- A cada passo pode simultaneamente consumir um caractere da entrada e um pedaço do topo da pilha

Autômato com pilha

Idéia:

- Controle finito, estado inicial, estados finais.
- Entrada: fita lida numa direção (como num AD).
- Memória auxiliar: pilha. `PUSH, POP, ISEMPTY`

Funcionamento:

- Começa no estado inicial
- A cada passo pode simultaneamente consumir um caractere da entrada e um pedaço do topo da pilha
- Muda de estado e empilha.

Autômato com pilha

Idéia:

- Controle finito, estado inicial, estados finais.
- Entrada: fita lida numa direção (como num AD).
- Memória auxiliar: pilha. `PUSH, POP, ISEMPTY`

Funcionamento:

- Começa no estado inicial
- A cada passo pode simultaneamente consumir um caractere da entrada e um pedaço do topo da pilha
- Muda de estado e empilha.
- Fim da entrada: aceita se estado é final (e pilha vazia).

Autômato com pilha - PDA

Autômato com pilha - PDA

$\mathscr{A} = (K, \Sigma, \Gamma, \Delta, s, F)$

- K é o conjunto de estados, finito,
- Σ é um alfabeto (de entrada),
- Γ é um alfabeto (da pilha),
- $s \in K$ é o estado inicial,
- $F \subseteq K$ são os estados finais,
- $\Delta \subset (K \times \Sigma \cup \{\lambda\} \times \Gamma^*) \times (K \times \Gamma^*)$ é um conjunto *finito* de transições.

AP

Autômato com pilha - PDA

$\mathscr{A} = (K, \Sigma, \Gamma, \Delta, s, F)$

- K é o conjunto de estados, finito,
- Σ é um alfabeto (de entrada),
- Γ é um alfabeto (da pilha),
- $s \in K$ é o estado inicial,
- $F \subseteq K$ são os estados finais,
- $\Delta \subset (K \times \Sigma \cup \{\lambda\} \times \Gamma^*) \times (K \times \Gamma^*)$ é um conjunto *finito* de transições.

Transição $(p, \sigma, \alpha), (q, \beta) \in \Delta$ diz:
se está em p, tem σ na entrada e α no topo da pilha, mude para q e substitua α por β no topo da pilha

Configurações

Configurações

Uma configuração de $\mathscr{A}$ é um elemento de $K \times \Sigma^* \times \Gamma^*$, a ser entendido como (estado atual, fita de entrada, pilha).

Configurações

Uma configuração de $\mathscr{A}$ é um elemento de $K \times \Sigma^* \times \Gamma^*$, a ser entendido como (estado atual, fita de entrada, pilha).

Dizemos que (p, x, α) produz em um passo (q, y, ζ)
Notação: $(p, x, \alpha) \vdash_{\mathscr{A}} (q, y, \zeta)$.

Configurações

Uma configuração de $\mathscr{A}$ é um elemento de $K \times \Sigma^* \times \Gamma^*$, a ser entendido como (estado atual, fita de entrada, pilha).

Dizemos que (p, x, α) produz em um passo (q, y, ζ)
Notação: $(p, x, \alpha) \vdash_{\mathscr{A}} (q, y, \zeta)$.

produz: fecho reflexivo transitivo de $\vdash_{\mathscr{A}}$, denotado $\vdash_{\mathscr{A}}^*$.

Configurações

Uma configuração de $\mathscr{A}$ é um elemento de $K \times \Sigma^* \times \Gamma^*$, a ser entendido como (estado atual, fita de entrada, pilha).

Dizemos que (p, x, α) produz em um passo (q, y, ζ)
Notação: $(p, x, \alpha) \vdash_{\mathscr{A}} (q, y, \zeta)$.

produz: fecho reflexivo transitivo de $\vdash_{\mathscr{A}}$, denotado $\vdash_{\mathscr{A}}^*$.

$\mathscr{A}$ aceita $x \in \Sigma^*$ se $(s, x, \lambda) \vdash_{\mathscr{A}}^* (p, \lambda, \lambda)$, para algum $p \in F$.
$L(\mathscr{A})$ é o conjunto de palavras aceitas por $\mathscr{A}$.

Exemplo (LP 3.3.1)

Exemplo (LP 3.3.1)

$L = \{wcw^R \mid w \in \{a, b\}^*\}$

Exemplo (LP 3.3.1)

$L = \{wcw^R \mid w \in \{a,b\}^*\}$

$\mathscr{A}$: $K = \{s, f\}$, $\Sigma = \{a, b, c\}$, $\Gamma = \{a, b\}$, $F = \{f\}$,

$$\Delta = \begin{cases} (1) & ((s,a,\lambda),(s,a)) \\ (2) & ((s,b,\lambda),(s,b)) \\ (3) & ((s,c,\lambda),(f,\lambda)) \\ (4) & ((f,a,a))(f,\lambda)) \\ (5) & ((f,b,b),(f,\lambda)) \end{cases}$$

	Entr	Pilha	Est
	abbcbba	λ	s
1	bbcbba	a	s
2	bcbba	ba	s
2	cbba	bba	s
3	bba	bba	f
5	ba	ba	f
5	a	a	f
4	λ	λ	f ✓

Exemplo (LP 3.3.2)

Exemplo (LP 3.3.2)

$L = \{ww^R \mid w \in \{a, b\}^*\}$

Exemplo (LP 3.3.2)

$L = \{ww^R \mid w \in \{a,b\}^*\}$

$\mathscr{A}$: $K = \{s, f\}$, $\Sigma = \{a, b\}$, $\Gamma = \{a, b\}$, $F = \{f\}$,

$$\Delta = \begin{cases} (1) & ((s,a,\lambda),(s,a)) \\ (2) & ((s,b,\lambda),(s,b)) \\ (3) & ((s,\lambda,\lambda),(f,\lambda)) \\ (4) & ((f,a,a))(f,\lambda)) \\ (5) & ((f,b,b),(f,\lambda)) \end{cases}$$

Regra	Ent	Pilha	Est
	abba	λ	s
1	bba	a	s
2	ba	ba	s
3	ba	ba	f
5	a	a	f
4	λ	λ	f ✓

Reconhecimento de LC

Teorema

Uma linguagem é reconhecida por um autômato com pilha sse ela é livre de contexto.

Reconhecimento de LC

Teorema

Uma linguagem é reconhecida por um autômato com pilha sse ela é livre de contexto.

Reconhecimento de LC

Teorema

Uma linguagem é reconhecida por um autômato com pilha sse ela é livre de contexto.

Demonstrações construtivas.

Reconhecimento de LC

Teorema

Uma linguagem é reconhecida por um autômato com pilha sse ela é livre de contexto.

Demonstrações construtivas.

Mais importante: construir um AP dada uma gramática — existem várias construções.

Reconhecimento de LC

Teorema

Uma linguagem é reconhecida por um autômato com pilha sse ela é livre de contexto.

Demonstrações construtivas.

Mais importante: construir um AP dada uma gramática — existem várias construções.

Fato: não-determinismo é necessário para o teorema.

Reconhecimento de LC

Teorema

Uma linguagem é reconhecida por um autômato com pilha sse ela é livre de contexto.

Demonstrações construtivas.

Mais importante: construir um AP dada uma gramática — existem várias construções.

Fato: não-determinismo é necessário para o teorema.

Fato: não-determinismo é ruim na prática.

Autômato com pilha determinístico

Autômato com pilha determinístico

Transições $((p,\sigma,\alpha),(\cdot,\cdot))$ e $((p,\tau,\beta),(\cdot,\cdot))$ são *compatíveis* se em cada par $(\sigma,\tau),(\alpha,\beta)$ uma das componentes é prefixo da outra.

Autômato com pilha determinístico

Transições $((p, \sigma, \alpha), (\cdot, \cdot))$ e $((p, \tau, \beta), (\cdot, \cdot))$ são *compatíveis* se em cada par $(\sigma, \tau), (\alpha, \beta)$ uma das componentes é prefixo da outra.

Um autômato com pilha é determinístico se não tem transições distintas compatíveis.

Autômato com pilha determinístico

Transições $((p,\sigma,\alpha),(\cdot,\cdot))$ e $((p,\tau,\beta),(\cdot,\cdot))$ são *compatíveis* se em cada par $(\sigma,\tau),(\alpha,\beta)$ uma das componentes é prefixo da outra.

Um autômato com pilha é determinístico se não tem transições distintas compatíveis.

Num APD, cada configuração produz no máximo uma outra em 1 passo.

Autômato com pilha determinístico

Transições $((p,\sigma,\alpha),(\cdot,\cdot))$ e $((p,\tau,\beta),(\cdot,\cdot))$ são *compatíveis* se em cada par $(\sigma,\tau),(\alpha,\beta)$ uma das componentes é prefixo da outra.

Um autômato com pilha é determinístico se não tem transições distintas compatíveis.

Num APD, cada configuração produz no máximo uma outra em 1 passo.

Uma linguagem LC L é determinística se existe um APD determinístico que reconhece $L\$$, onde $\$ \notin \Sigma$ é um marcador de fim da entrada.

A arte do determinismo

A arte do determinismo

Existem vários truques para modificar uma gramática, de forma que, se der tudo certo, o AP construído é determinístico.

A arte do determinismo

Existem vários truques para modificar uma gramática, de forma que, se der tudo certo, o AP construído é determinístico.

Objetivo: dada uma palavra de L, construir uma árvore de derivação (*análise sintática, parsing*).

Um compilador típico

Para uma linguagem dada pro uma gramática LC.

- Fase 1: Analisador léxico (`flex`); expressões regulares descrevem os ítens, analisador tokeniza o texto. Token: x que é ou uma letra no alfabeto da linguagem ou tal que exista alguma $T \Rightarrow_G^* x$.

Um compilador típico

Para uma linguagem dada pro uma gramática LC.

- Fase 1: Analisador léxico (`flex`); expressões regulares descrevem os ítens, analisador tokeniza o texto. Token: x que é ou uma letra no alfabeto da linguagem ou tal que existista alguma $T \Rightarrow_G^* x$.
- Fase 2: Análise sintática. Produz a árvore de derivação (`bison`).

Um compilador típico

Para uma linguagem dada pro uma gramática LC.

- Fase 1: Analisador léxico (`flex`); expressões regulares descrevem os ítens, analisador tokeniza o texto. Token: x que é ou uma letra no alfabeto da linguagem ou tal que exista alguma $T \Rightarrow_G^* x$.
- Fase 2: Análise sintática. Produz a árvore de derivação (`bison`).
- Fase 3: Geração de código. Otimização

Um compilador típico

Para uma linguagem dada pro uma gramática LC.

- Fase 1: Analisador léxico (`flex`); expressões regulares descrevem os ítens, analisador tokeniza o texto. Token: x que é ou uma letra no alfabeto da linguagem ou tal que exista alguma $T \Rightarrow_G^* x$.
- Fase 2: Análise sintática. Produz a árvore de derivação (`bison`).
- Fase 3: Geração de código. Otimização
- Fase 4: Link-edição.

Um compilador típico

Para uma linguagem dada pro uma gramática LC.

- Fase 1: Analisador léxico (`flex`); expressões regulares descrevem os ítens, analisador tokeniza o texto. Token: x que é ou uma letra no alfabeto da linguagem ou tal que exista alguma $T \Rightarrow_G^* x$.
- Fase 2: Análise sintática. Produz a árvore de derivação (`bison`).
- Fase 3: Geração de código. Otimização
- Fase 4: Link-edição.
- Frequentemente algumas dessas fases ocorrem simultaneamente, como pipeline.