Análise Sintática

Linguagens Formais e Autómatos

Francisco Coelho
fc@di.uevora.pt

Departamento de Informática
Escola de Ciências e Tecnologia
Universidade de Évora

UNIVERSIDADE DE ÉVORA

UNIVERSIDADE DE ÉVORA

Análise Sintáctica

Limpeza de uma Gramática

Tipos de Análise Sintáctica

UNIVERSIDADE DE ÉVORA

Problema: Dada uma GIC G e uma palavra w, como determinar se $w \in \mathcal{L}(G)$? E, em caso afirmativo, também obter uma derivação de w.

É preciso pesquisar (a árvore d)as possíveis derivações...

- sentido
  - descendente a partir do símbolo inicial da gramática;
  - ascendente a partir da palavra;
- estratégia
  - em largura expandir primeiro todos os sub-ramos;
  - em profundidade desenvolver completamente cada ramo;

O prefixo terminal de $w = uAv, A \in V, u \in \Sigma^*$ é u, o maior prefixo de w formado apenas por terminais.

Grafo de uma Gramática

UNIVERSIDADE DE ÉVORA

O grafo esquerdo da GIC $G = (V, \Sigma, P, S)$ é o digrafo etiquetado $\mathcal{G}_L(G)$ onde

nós

$$N = \{w \in (V \cup \Sigma)^* \ : \ S \Rightarrow_L^* w\}$$

arcos

$$A = \{[u, v, p] \in N \times N \times P \ : \ u \Rightarrow_L v \text{ por } p\}$$

O grafo direito define-se de modo análogo.

O grafo esquerdo (ou direito) de uma gramática não ambígua é uma árvore.

Exemplo — Grafo Esquerdo

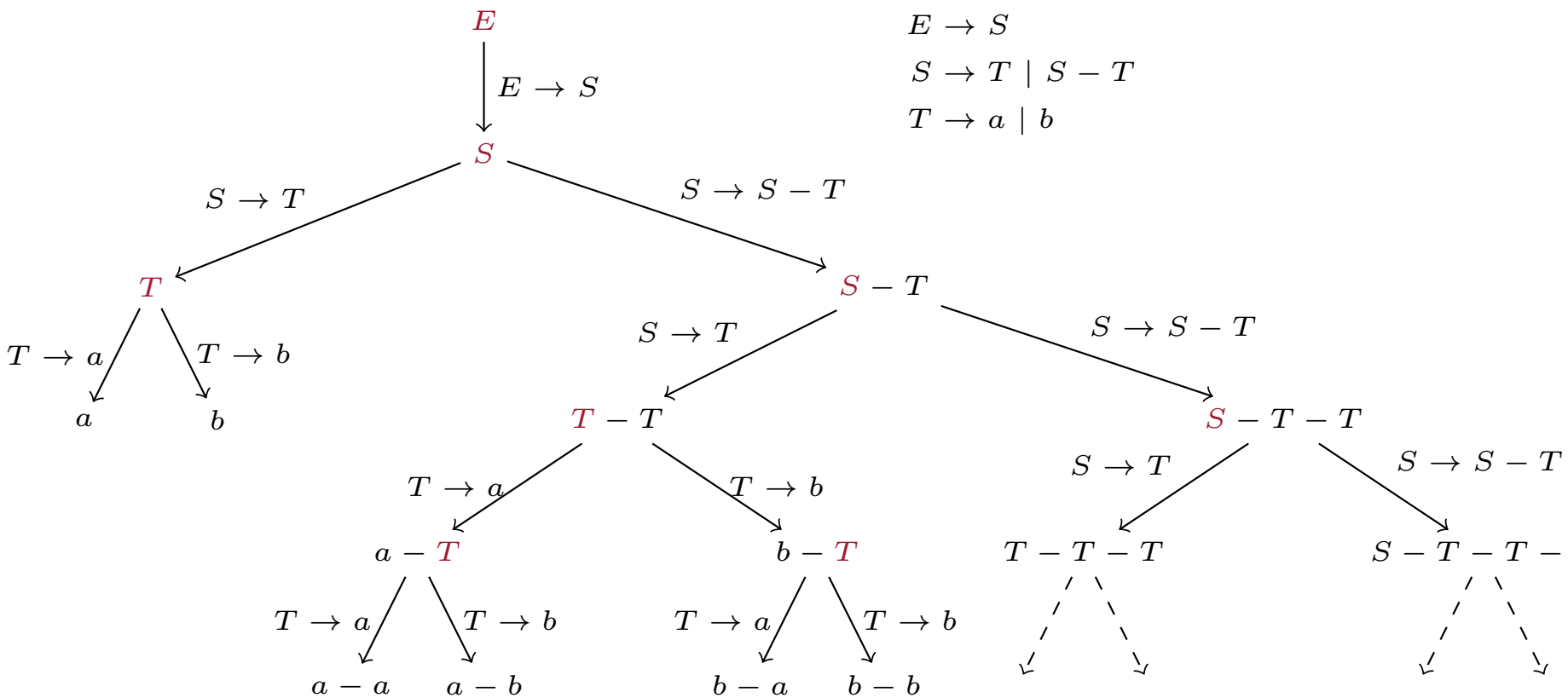
UNIVERSIDADE DE ÉVORA

$$ES = (\{E, S, T\}, \{a, b, -\}, \ldots, E)$$

$$E \to S$$

$$S \to T \mid S - T$$

$$T \to a \mid b$$



Exemplo — Grafo Direito

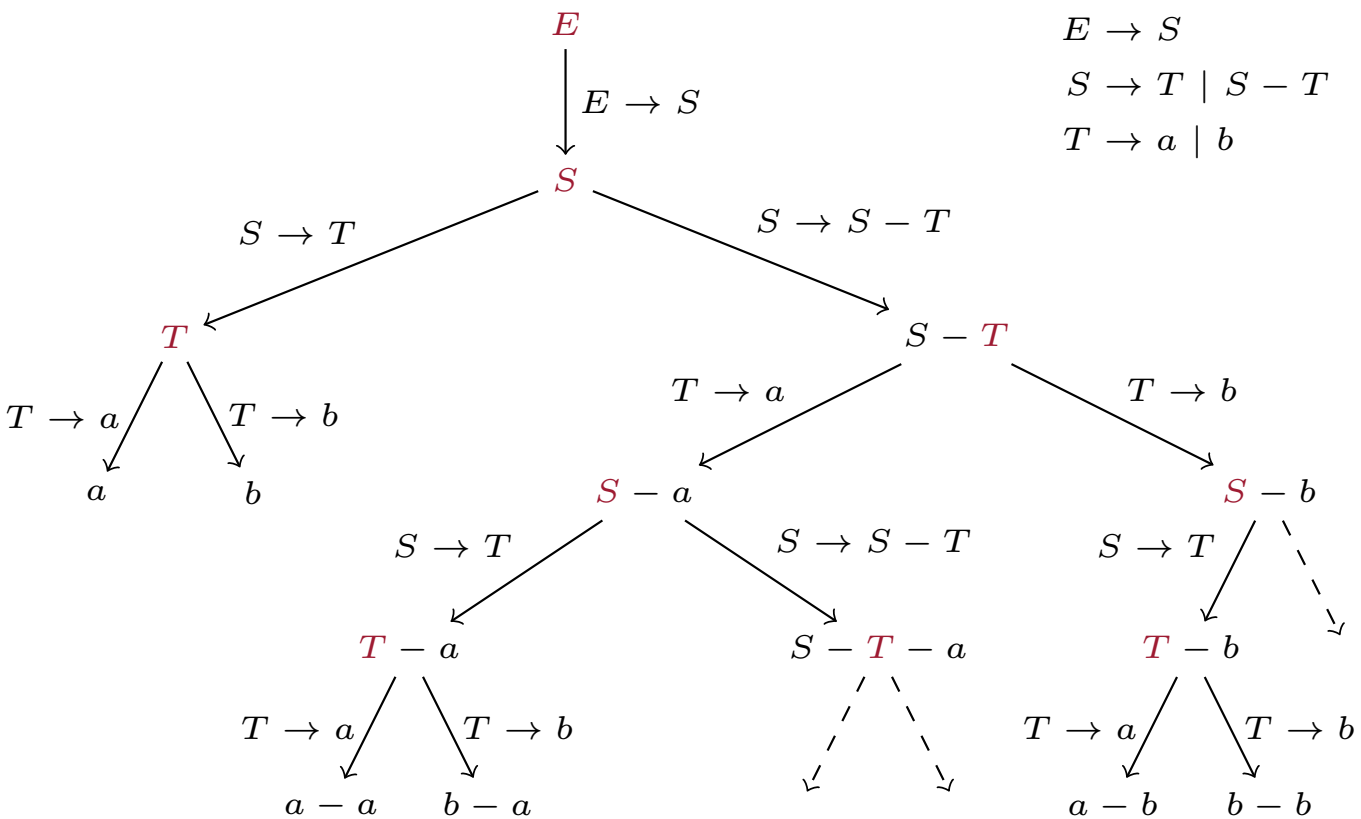
UNIVERSIDADE DE ÉVORA

$$ES = (\{E, S, T\}, \{a, b, -\}, \ldots, E)$$

$$E \to S$$

$$S \to T \mid S - T$$

$$T \to a \mid b$$



Algoritmo para AS Descendente em Largura

UNIVERSIDADE DE ÉVORA

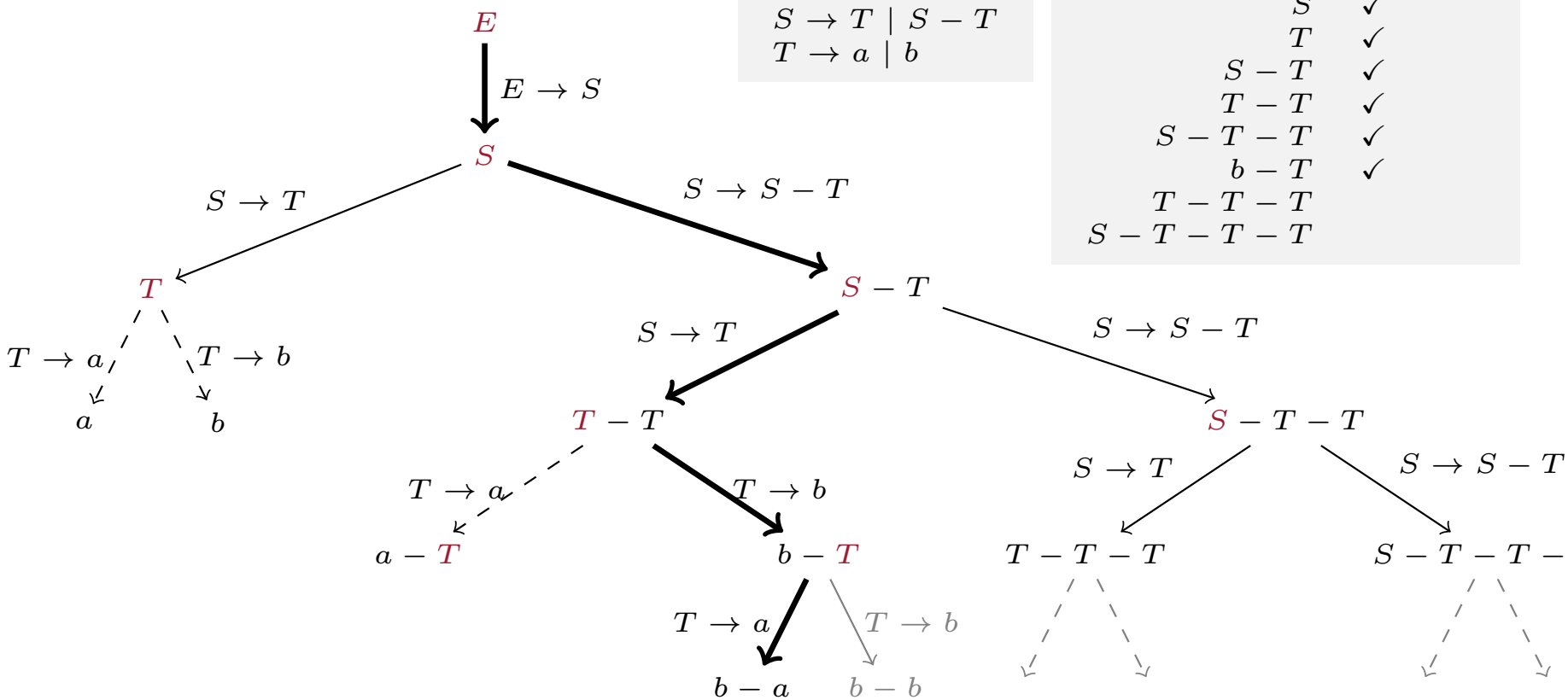
entrada: GIC $G = (V, \Sigma, P, S)$
entrada: palavra $p \in \Sigma^*$
    cria árvore T com raiz S ▷ árvore de pesquisa
    $Q \leftarrow \{S\}$ ▷ fila (FIFO)
    repete
        $q \leftarrow$ remove(Q) ▷ nó a expandir
        $i \leftarrow 0$ ▷ número da última produção usada
        feito $\leftarrow$ falso ▷ expansão completa?
        $q = uAv$ ▷ u é o prefixo terminal de q
        repete ▷ gerar todos os filhos de q
            se não há produção para A com número maior que i **então**
                feito $\leftarrow$ verdade
            senão ▷ acrescentar todos os filhos $uAv \Rightarrow uwv$
                seja $A \rightarrow w$ a primeira produção com número $j > i$
                se $uwv \notin \Sigma^*$ e o prefixo terminal de uwv é um prefixo de p **então**
                    insere uwv em Q
                    acrescenta o nó uwv a T como filho de q
                $i \leftarrow j$
        até feito **ou** $p = uwv$
    até vazia(Q) **ou** $p = uwv$
    se $p = uwv$ **então**
        ACEITA
    senão
        REJEITA

Exemplo — ASDL para $b - a$

UNIVERSIDADE DE ÉVORA

$E \to S$
$S \to T \mid S - T$
$T \to a \mid b$

fila	removido
E	✓
S	✓
T	✓
$S - T$	✓
$T - T$	✓
$S - T - T$	✓
$b - T$	✓
$T - T - T$	
$S - T - T - T$	



UNIVERSIDADE DE ÉVORA

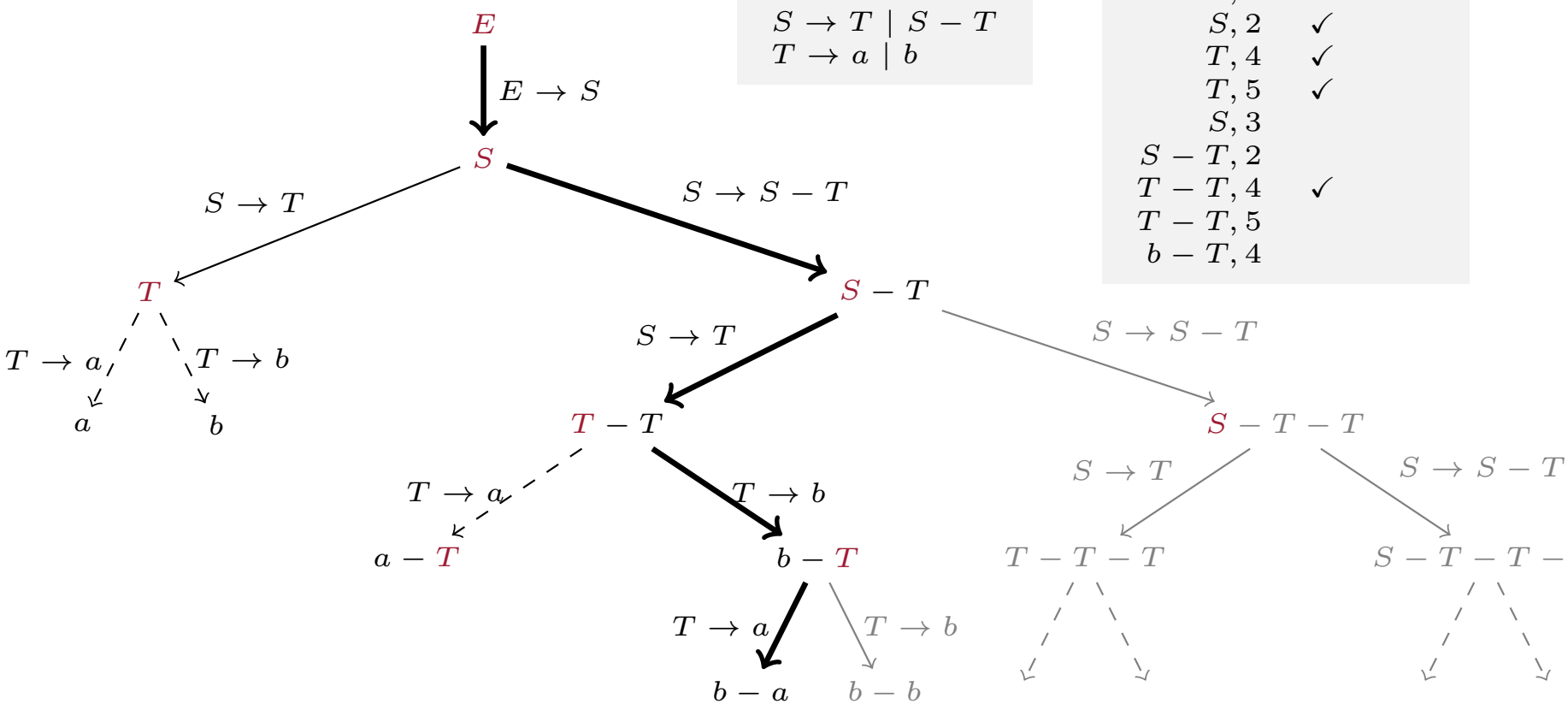
entrada: GIC $G = (V, \Sigma, P, S)$
entrada: palavra $p \in \Sigma^*$
    $X \leftarrow \{[S, 0]\}$ ▷ pilha (LIFO)
    repete
        $[q, i] \leftarrow$ desempilha(X)
        inviável $\leftarrow$ falso
        repete
            $q = uAv$ com $u \in \Sigma^*$ e $A \in V$
            se u não é prefixo de p **então**
                inviável $\leftarrow$ verdade
            se não há produção para A com número maior que i **então**
                inviável $\leftarrow$ verdade
            se não inviável **então**
                seja $A \rightarrow w$ a primeira produção para A com número $j > i$
                empilha$([q, j], X)$
                $q \leftarrow uwv$
                $i \leftarrow 0$
        até inviável **ou** $q \in \Sigma^*$
    até vazia(X) **ou** $q = p$
    se $q = p$ **então**
        ACEITA
    senão
        REJEITA

Exemplo — ASDP para $b - a$

UNIVERSIDADE DE ÉVORA

$$E \to S$$
$$S \to T \mid S - T$$
$$T \to a \mid b$$

pilha	removido
$E, 0$	✓
$E, 1$	
$S, 2$	✓
$T, 4$	✓
$T, 5$	✓
$S, 3$	
$S - T, 2$	
$T - T, 4$	✓
$T - T, 5$	
$b - T, 4$	



UNIVERSIDADE DE ÉVORA

- Parte da palavra e pesquisa um caminho para o símbolo inicial para evitar os ramos desnecessários que ocorrem na pesquisa descende;
- Usa

  Transferência mover símbolos da entrada para a "pilha de trabalho";

  Redução substituir uma sub-palavra w por uma variável A se houver uma produção $A \to w$. Isto é, reverter a aplicação da produção $A \to w$;

Exemplo — A.S. Ascendente "à mão"

UNIVERSIDADE DE ÉVORA

Gramática

$$E \to S$$
$$S \to T \mid S - T$$
$$T \to a \mid b$$

Rascunho

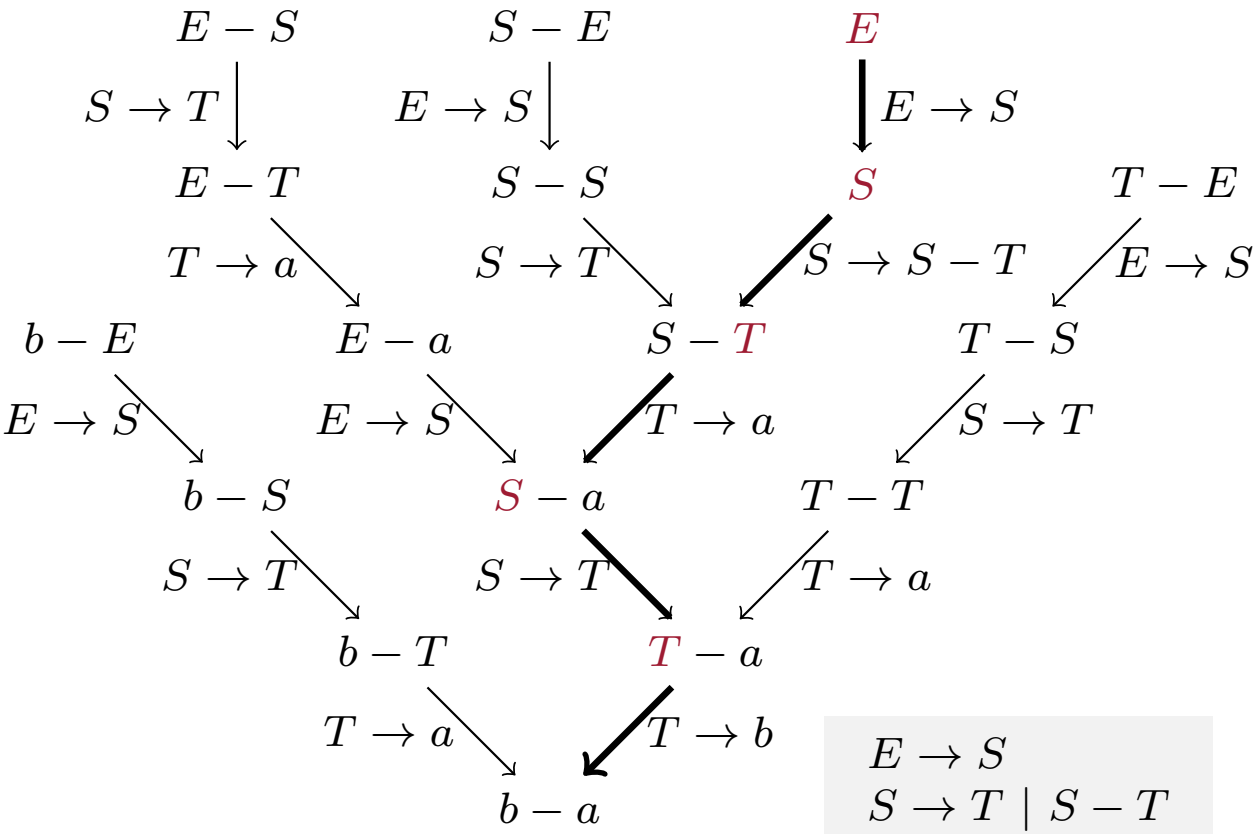
pilha	entrada	operação	
$\$\lambda$	$b - a\#$	TRANSFERE	
$\$b$	$- a\#$	REDUZ	$T \to b$
$\$T$	$- a\#$	REDUZ	$S \to T$
$\$S$	$- a\#$	TRANSFERE	
$\$S -$	$a\#$	TRANSFERE	
$\$S - a$	$\lambda\#$	REDUZ	$T \to a$
$\$S - T$	$\lambda\#$	REDUZ	$S \to S - T$
$\$S$	$\lambda\#$	REDUZ	$E \to S$
$\$E$	$\lambda\#$	ACEITA	

Derivação: $E \Rightarrow S \Rightarrow S - T \Rightarrow S - a \Rightarrow T - a \Rightarrow b - a$

UNIVERSIDADE DE ÉVORA

entrada: GIC $G = (V, \Sigma, P, S)$
entrada: $p \in \Sigma^*$
  cria árvore T com raiz p ▷ árvore de pesquisa
  $Q \leftarrow \{p\}$ ▷ fila (FIFO)
  repete
    $q \leftarrow$ remove(Q)
    para cada produção $A \rightarrow w \in P$
      para cada decomposição $q = uwv$ com $v \in \Sigma^*$ ▷ TRANSF.
        insere(uAv, Q) ▷ REDUÇÃO
        acrescenta uAv aos filhos de q
  até $q = S$ **ou** vazia(Q)
  se $q = S$ **então**
    ACEITA
  senão
    REJEITA

Exemplo — A.S. Ascendente em Largura para $b\ a$



$$E \to S$$
$$S \to T \mid S - T$$
$$T \to a \mid b$$

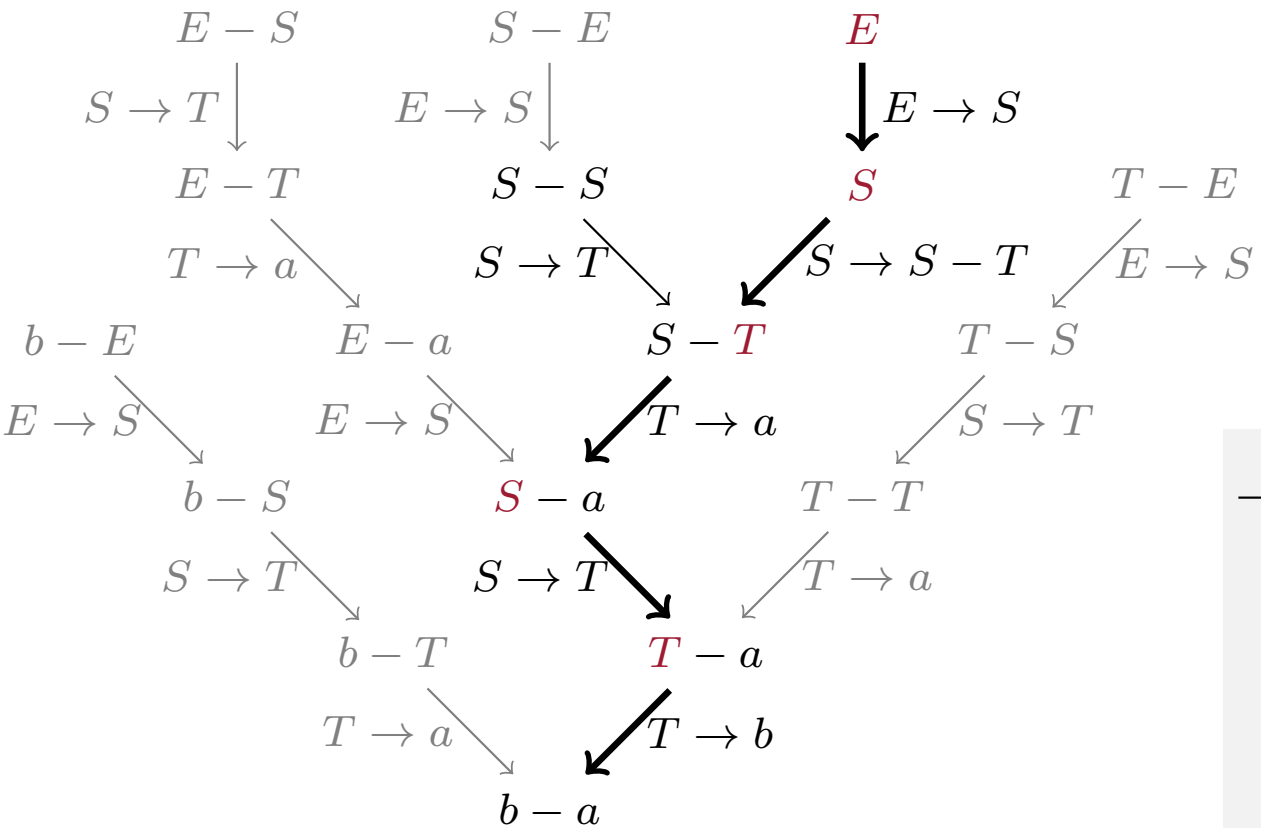
fila	removido
$b - a$	✓
$b - T$	✓
$T - a$	✓
$b - S$	✓
$S - a$	✓
$T - T$	✓
$b - E$	✓
$E - a$	✓
$S - T$	✓
$T - S$	✓
$E - T$	✓
$S - S$	✓
S	✓
$T - E$	
$E - S$	
$S - E$	

Análise Sintática Ascendente em Profundidade

UNIVERSIDADE DE ÉVORA

entrada: GIC $G = (V, \Sigma, P, S)$ com S não recursivo
entrada: palavra $p \in \Sigma^*$
    $X \leftarrow \{[\lambda, p, 0]\}$ ▷ pilha (LIFO)
    repete
        $[u, v, i] \leftarrow$ desempilha(X)
        inviável $\leftarrow$ falso
        repete
            seja $j > i$ o número da primeira produção da forma
                $A \to w$ com $u = qw$ e $A \neq S$ ou
                $S \to w$ com $u = w$ e $v = \lambda$
            se existe tal j então
                empilha$([u, v, j], X)$
                $u \leftarrow qA$ ▷ REDUÇÃO
                $i \leftarrow 0$
            se não existe tal j e $v \neq \lambda$ então
                TRANSFERÊNCIA(u, v)
                $i \leftarrow 0$
            se não existe tal j e $v = \lambda$ então
                inviável $\leftarrow$ verdade
        até $u = S$ ou inviável
    até $u = S$ ou vazia(X)
    se vazia(X) então
        REJEITA
    senão
        ACEITA

Exemplo — ASAP para $b - a$



$E \to S$
$S \to T \mid S - T$
$T \to a \mid b$

pilha	removido
$\lambda, b - a, 0$	✓
$b, -a, 5$	
$T, -a, 2$	
$S - a, \lambda, 4$	
$S - T, \lambda, 2$	✓
$S - T, \lambda, 3$	
$S, \lambda, 1$	

UNIVERSIDADE DE ÉVORA

Análise Sintáctica

Limpeza de uma Gramática

O bom funcionamento dos algoritmos anteriores depende da gramática usada.

Problemas como...

- Pesquisas infinitas, quando a palavra não é derivada pela gramática;
- Exploração de ramos repetidos, quando a gramática é ambígua;

reduzem o desempenho de um analizador sintáctico.

Mas é possível, em certos casos, "limpar" a gramática de forma a aliviar esses problemas.

Símbolo Inicial Não Recursivo

UNIVERSIDADE DE ÉVORA

Para cada GIC $G = (V, \Sigma, P, S)$ existe uma GIC equivalente $G' = (V', \Sigma, P', S')$ onde o símbolo inicial não é recursivo:

- Se o símbolo inicial de G não é recursivo, $G' = G$;
- Se o símbolo inicial de G é recursivo:

$$G' = \left(V \cup \left\{S'\right\}, \Sigma, P \cup \left\{S' \to S\right\}, S'\right) \quad \text{com } S' \notin V$$

Exemplo — Tornar o Símbolo Inicial Não Recursivo

UNIVERSIDADE DE ÉVORA

Gramática original com o símbolo inicial, L, recursivo

$$G = (\{L, M, N, O\}, \{a, b, c, d\}, \ldots, L)$$
$$L \to Mb \mid aLb \mid \lambda$$
$$M \to Lb \mid MLN \mid \lambda$$
$$N \to NaN \mid NbO$$
$$O \to cO \mid \lambda$$

Gramática equivalente com o símbolo inicial, L', não recursivo

$$G' = (\{L', L, M, N, O\}, \{a, b, c, d\}, \ldots, L')$$
$$L' \to L$$
$$L \to Mb \mid aLb \mid \lambda$$
$$M \to Lb \mid MLN \mid \lambda$$
$$N \to NaN \mid NbO$$
$$O \to cO \mid \lambda$$

Seja $G = (V, \Sigma, P, S)$ uma gramática independente do contexto.

- Uma produção-λ é uma produção da forma $A \to \lambda$;
- Uma produção unitária é uma produção da forma $A \to B$;
- O conjunto dos símbolos que geram λ é

$$\Lambda = \{A \in V \ : \ A \Rightarrow^* \lambda\}$$

- Numa gramática não contraível não existem símbolos que geram λ;
- Numa gramática essencialmente não contraível só o símbolo inicial pode gerar λ;

Portanto, numa derivação de uma gramática essencialmente não contraível, os passos intermédios só podem diminuir de tamanho por aplicação da regra $S \to \lambda$.

Seja $G = (V, \Sigma, P, S)$ uma gramática independente do contexto.

Se $A \Rightarrow_G^* u$ então $G' = (V, \Sigma, P \cup \{A \to u\}, S)$ é equivalente a G.

Se $A \to uBv, B \to b_1 | \cdots | b_n \in P$ então

$$G' = (V, \Sigma, P \setminus \{A \to uBv\} \cup \{A \to ub_1v | \cdots | ub_nv\}, S)$$

é equivalente a G.

Eliminação de Produções Vazias

UNIVERSIDADE DE ÉVORA

Seja $G = (V, \Sigma, P, S)$ uma GIC com S não recursivo.

A gramática $G_L = (V, \Sigma, P_L, S)$ cujas produções são, em conjunto

- Todas as produções de G que não são produções-λ;
- Todas as produções que se obtêm eliminando um ou mais símbolos de Λ do corpo de uma produção de P desde que o corpo que resulta tenha pelo menos um símbolo;
- A produção $S \to \lambda$ se $S \in \Lambda$;

é equivalente a G e essencialmente não contraível.

Exemplo — Eliminar produções vazias

UNIVERSIDADE DE ÉVORA

Gramática com o símbolo inicial, L', não recursivo

$$
\begin{aligned}
G' &= \left(\left\{L', L, M, N, O\right\}, \{a, b, c, d\}, \ldots, L'\right) \\
L' &\to L \\
L &\to Mb \mid aLb \mid \lambda \\
M &\to Lb \mid MLN \mid \lambda \\
N &\to NaN \mid NbO \\
O &\to cO \mid \lambda
\end{aligned}
$$

Símbolos que geram λ: $\Lambda = \{L', L, M, O\}$

Gramática equivalente e essencialmente não contraível

$$
\begin{aligned}
G'' &= \left(\left\{L', L, M, N, O\right\}, \{a, b, c, d\}, \ldots, L'\right) \\
L' &\to L \mid \lambda \\
L &\to Mb \mid aLb \mid b \mid ab \\
M &\to Lb \mid MLN \mid b \mid LN \mid MN \mid N \\
N &\to NaN \mid NbO \mid Nb \\
O &\to cO \mid c
\end{aligned}
$$

Seja $G = (V, \Sigma, P, S)$ uma GIC essencialmente não contraível.

Para cada $A \in V$ seja

$$\text{CHAIN}(A) = \{B \in V \ : \ A \Rightarrow_G^* B\}$$

A gramática $G' = (V, \Sigma, \ldots, S)$ cujas produções $A \to w$ satisfazem

1. $w \notin V$ **e**
2. existe algum $B \in \text{CHAIN}(A)$ tal que $B \to w \in P$

é equivalente a G.

Exemplo — Eliminar produções unitárias

UNIVERSIDADE DE ÉVORA

Gramática essencialmente não contraível

$$G'' = \left(\left\{L', L, M, N, O\right\}, \{a, b, c, d\}, \ldots, L'\right)$$

$$\begin{aligned}
L' &\to L \mid \lambda \\
L &\to Mb \mid aLb \mid b \mid ab \\
M &\to Lb \mid MLN \mid b \mid LN \mid MN \mid N \\
N &\to NaN \mid NbO \mid Nb \\
O &\to cO \mid c
\end{aligned}$$

$$\begin{aligned}
\mathsf{CHAIN}(L') &= \{L', L\} \\
\mathsf{CHAIN}(L) &= \{L\} \\
\mathsf{CHAIN}(M) &= \{M, N\} \\
\mathsf{CHAIN}(N) &= \{N\} \\
\mathsf{CHAIN}(O) &= \{O\}
\end{aligned}$$

Gramática equivalente e sem produções unitárias

$$G''' = \left(\left\{L', L, M, N, O\right\}, \{a, b, c, d\}, \ldots, L'\right)$$

$$\begin{aligned}
L' &\to Mb \mid aLb \mid b \mid ab \mid \lambda \\
L &\to Mb \mid aLb \mid b \mid ab \\
M &\to Lb \mid MLN \mid b \mid LN \mid MN \mid NaN \mid NbO \mid Nb \\
N &\to NaN \mid NbO \mid Nb \\
O &\to cO \mid c
\end{aligned}$$

Símbolos Inúteis

UNIVERSIDADE DE ÉVORA

Seja $G = (V, \Sigma, P, S)$ uma gramática independente do contexto.

Um símbolo $x \in V \cup \Sigma$ é útil se existe uma derivação

$$S \Rightarrow^* uxv \Rightarrow^* w \quad \text{com } u, v \in (V \cup \Sigma)^*, w \in \Sigma^*$$

- um símbolo que não é útil é inútil
- um símbolo, $x \in V \cup \Sigma$, é acessível se $S \Rightarrow^* uxv$ com $u, v \in (V \cup \Sigma)^*$
- um símbolo que não é acessível é inacessível
- um símbolo não terminal, A, é produtivo se $A \Rightarrow^* w$ com $w \in \Sigma^*$
- um símbolo não terminal que não é produtivo é improdutivo
- Um símbolo não terminal é útil se e só se for acessível e produtivo

Limpeza de uma Gramática

UNIVERSIDADE DE ÉVORA

1. Eliminar os não terminais improdutivos:
   1.1 Calcular os símbolos não terminais PRODUTIVOS;
   1.2 Remover as produções em que ocorrem não terminais não produtivos;
2. Eliminar os não terminais inacessíveis:
   2.1 Calcular os símbolos ACESSÍVEIS;
   2.2 Remover as produções dos não terminais inacessíveis;

Uma gramática sem símbolos inúteis diz-se limpa ou reduzida.

Exemplo — Limpeza de uma Gramática

UNIVERSIDADE DE ÉVORA

$$
\begin{aligned}
G''' &= (\{L', L, M, N, O\}, \{a, b, c, d\}, \ldots, L') \\
L' &\to Mb \mid aLb \mid b \mid ab \mid \lambda \\
L &\to Mb \mid aLb \mid b \mid ab \\
M &\to Lb \mid MLN \mid b \mid LN \mid MN \mid NaN \mid NbO \mid Nb \\
N &\to NaN \mid NbO \mid Nb \\
O &\to cO \mid c \\
\text{PRODUTIVOS} &= \{L', L, M, O\} \\
L' &\to Mb \mid aLb \mid b \mid ab \mid \lambda \\
L &\to Mb \mid aLb \mid b \mid ab \\
M &\to Lb \mid b \\
O &\to cO \mid c \\
\text{ACESSÍVEIS} &= \{L', L, M, a, b\} \\
G^{iv} &= (\{L', L, M\}, \{a, b\}, \ldots, L') \\
L' &\to Mb \mid aLb \mid b \mid ab \mid \lambda \\
L &\to Mb \mid aLb \mid b \mid ab \\
M &\to Lb \mid b
\end{aligned}
$$

Forma Normal de Chomsky

UNIVERSIDADE DE ÉVORA

A gramática independente do contexto $G = (V, \Sigma, P, S)$ está na forma normal de Chomsky se todas as produções estão numa das formas seguintes

$$A \to BC \qquad B, C \in V \setminus \{S\}$$
$$A \to a \qquad a \in \Sigma$$
$$S \to \lambda$$

As árvores das derivações de uma gramática na forma normal de Chomsky são binárias.

Exemplo — Construir a Forma Normal de Chomsky

UNIVERSIDADE DE ÉVORA

$$G^v = (\{L', L, M, A, B\}, \{a, b\}, \ldots, L')$$

$$L' \to MB \mid ALB \mid b \mid AB \mid \lambda$$

$$L \to MB \mid ALB \mid b \mid AB$$

$$M \to LB \mid b$$

$$A \to a$$

$$B \to b$$

$$G_{NC} = (\{L', L, M, A, B, X\}, \{a, b\}, \ldots, L')$$

$$L' \to MB \mid AX \mid b \mid AB \mid \lambda$$

$$L \to MB \mid AX \mid b \mid AB$$

$$M \to LB \mid b$$

$$A \to a$$

$$B \to b$$

$$X \to LB$$

Forma Normal de Greibach

UNIVERSIDADE DE ÉVORA

A gramática independente do contexto $G = (V, \Sigma, P, S)$ está na forma normal de Greibach se todas as produções estão numa das formas seguintes

$$A \to aA_1A_2 \cdots A_k \qquad A_i \in V \setminus \{S\}, i = 1, 2, \ldots, k$$
$$A \to a \qquad a \in \Sigma$$
$$S \to \lambda$$

A forma normal de Greibach é usada para

1. evitar recursões infinitas nos algoritmos de análise sintáctica;
2. guiar (com o primeiro símbolo) a escolha das regras;
3. produzir um autómato de pilha equivalente à gramática dada;

Construção da Forma Normal de Greibach

UNIVERSIDADE DE ÉVORA

Dada uma GIC $G = (V, \Sigma, P, S)$ na forma normal de Chomsky:

1. Define-se (arbitrariamente) uma ordem total para os não terminais de G onde S seja o primeiro símbolo;
2. Segue-se essa ordem para transformar todas as produções de modo que se

$$A \to Bw, \quad B \in V$$

   então $A < B$;
3. Segue-se essa ordem invertida para substituir as produções $A \to Bw$, com $B \in V$ por $A \to uw$ para todas as produções $B \to u$ de B;

Passos 1 e 2: arbitrar uma ordem dos símbolos não terminais e, seguindo essa ordem, transformar as produções $A \to Bw$ de forma que seja $A < B$;

Ordem: $L' < X < B < A < L < M$

$$
\begin{aligned}
L' &\to MB \mid AX \mid b \mid AB \mid \lambda \\
X &\to LB \\
B &\to b \\
A &\to a \\
L &\to MB \mid aX \mid b \mid aB \\
(M &\to LB \mid b) \\
M &\to MBB \mid aXB \mid bB \mid aBB \mid b
\end{aligned}
$$

. . . e agora?

Se A é um símbolo não terminal com pelo menos uma produção da forma

$$A \to Au$$

substituem-se as produções

$$A \to Au_1 \mid Au_2 \mid \cdots \mid Au_j \mid v_1 \mid v_2 \mid \cdots \mid v_k \quad u_i, v_i \in (V \cup \Sigma)^*$$

onde o primeiro símbolo dos v_i não é A pelas produções

$$A \to v_1 Z \mid v_2 Z \mid \cdots \mid v_k Z \mid v_1 \mid v_2 \mid \cdots \mid v_k$$
$$Z \to u_1 Z \mid u_2 Z \mid \cdots \mid u_j Z \mid u_1 \mid u_2 \mid \cdots \mid u_j$$

sendo Z um novo símbolo não terminal.

Passo 3: substituir, por ordem invertida, cada produções $A \to Bw$ pelas produções $A \to uw$ para cada $B \to u$

$$G_{NG} = \left(\left\{L', X, B, A, L, M, Z\right\}, \{a, b\}, \ldots, L'\right)$$

$$L' \to aXBZB \mid bBZB \mid aBBZB \mid bZB \mid aXBB \mid bBB \mid aBBB \mid$$
$$\mid aX \mid b \mid aB \mid \lambda$$

$$X \to aXBZBB \mid bBZBB \mid aBBZBB \mid bZBB \mid aXBBB$$
$$\mid bBBB \mid aBBBB \mid bBB \mid aXB \mid bB \mid aBB$$

$$B \to b$$

$$A \to a$$

$$L \to aXBZB \mid bBZB \mid aBBZB \mid bZB \mid aXBB \mid bBB \mid aBBB \mid$$
$$\mid aX \mid b \mid aB$$

$$M \to aXBZ \mid bBZ \mid aBBZ \mid bZ \mid aXB \mid bB \mid aBB \mid b$$

$$Z \to bBZ \mid bB$$

Construção de Formas Normais

UNIVERSIDADE DE ÉVORA

1. Tornar o símbolo inicial não recursivo;
2. Transformar numa gramática essencialmente não contraível;
3. Eliminar as produções unitárias;
4. Eliminar os símbolos inúteis
   - 4.1 Eliminar os símbolos improdutivos;
   - 4.2 Eliminar os símbolos inacessíveis;
5. Construir a forma normal de Chomsky
   - 5.1 Transformar as produções com mais do que um símbolo no corpo para a forma $A \to A_1 A_2 \cdots A_k$;
   - 5.2 Transformar as produções com mais do que um símbolo no corpo para a forma $A \to BC$;
6. Construir a forma normal de Greibach
   - 6.1 Ordenar os não terminais;
   - 6.2 Garantir que não há produções da forma $A \to Bw$ com $B < A$ (eliminando a recursividade directa à esquerda, se necessário);
   - 6.3 Garantir que as produções são da forma $A \to aw$ com $a \in \Sigma$;