Compiladores
(IF688)

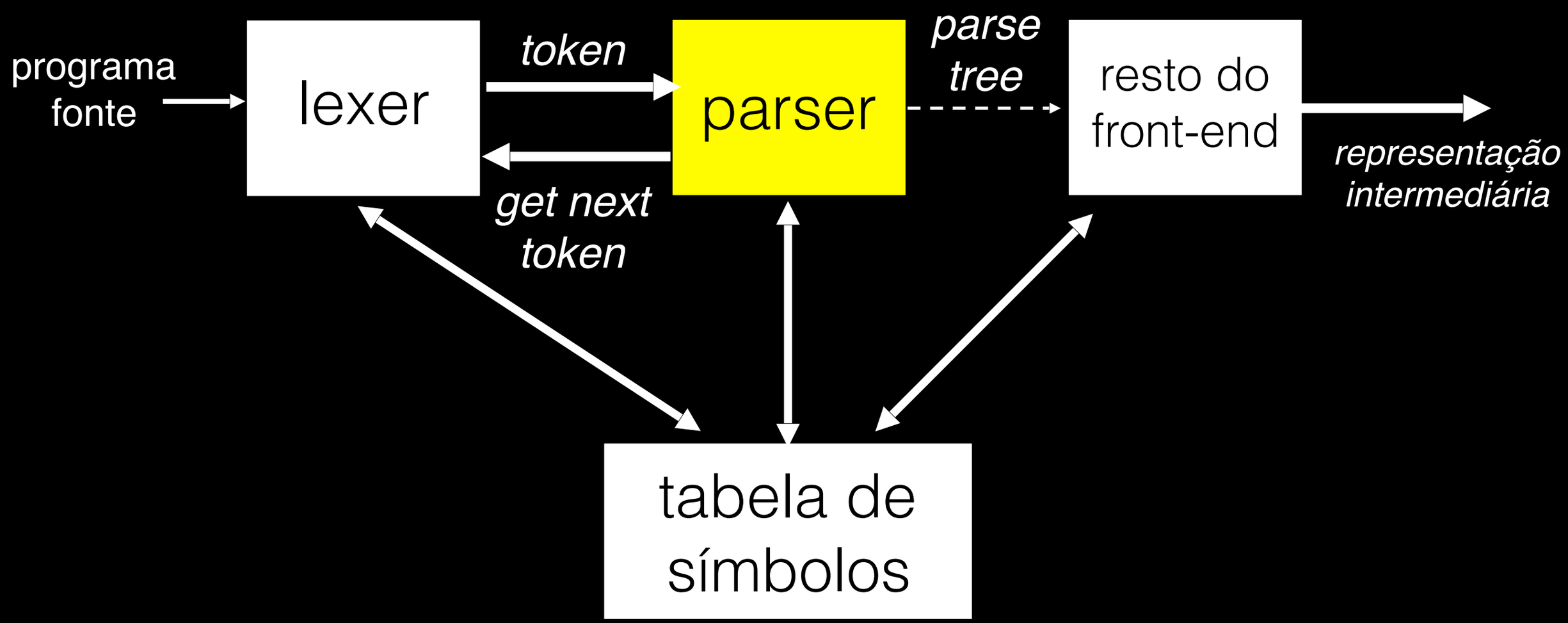
Leopoldo Teixeira
lmt@cin.ufpe.br | @leopoldomt

Contato

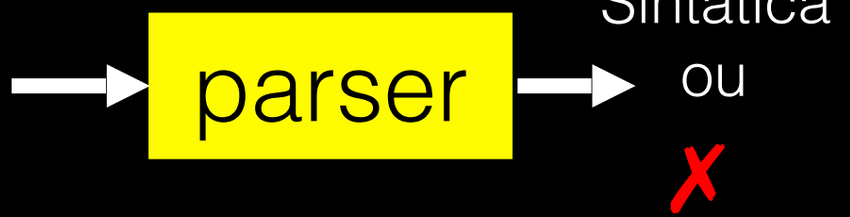
- Grupo de Discussão:
  https://classroom.google.com
  Faça o login com sua conta do CIn. Na página inicial, clique em +.
  Digite na caixa o código da turma: ***mpg9k1***
  Clique em PARTICIPAR

- E-mail: **lmt@cin.ufpe.br**
  Sala C012

Analisador Sintático



Parsing



- Caso sucesso:
  - A sintaxe do programa está correta
  - A string de entrada é “bem formada”
- Caso contrário:
  - erro de sintaxe; alguma regra sintática foi violada
- Importante:
  - O programa pode ainda conter erros capturados ou não pelo type checker

Gramáticas

- Especificações precisas e fáceis de entender da sintaxe de uma linguagem de programação
- Para algumas classes de gramáticas, é possível gerar automaticamente um parser eficiente.
- Tem relação direta com a estrutura da linguagem usada para tradução/compilação.
- Fáceis de serem estendidas e atualizadas.

Gramáticas Livres de Contexto

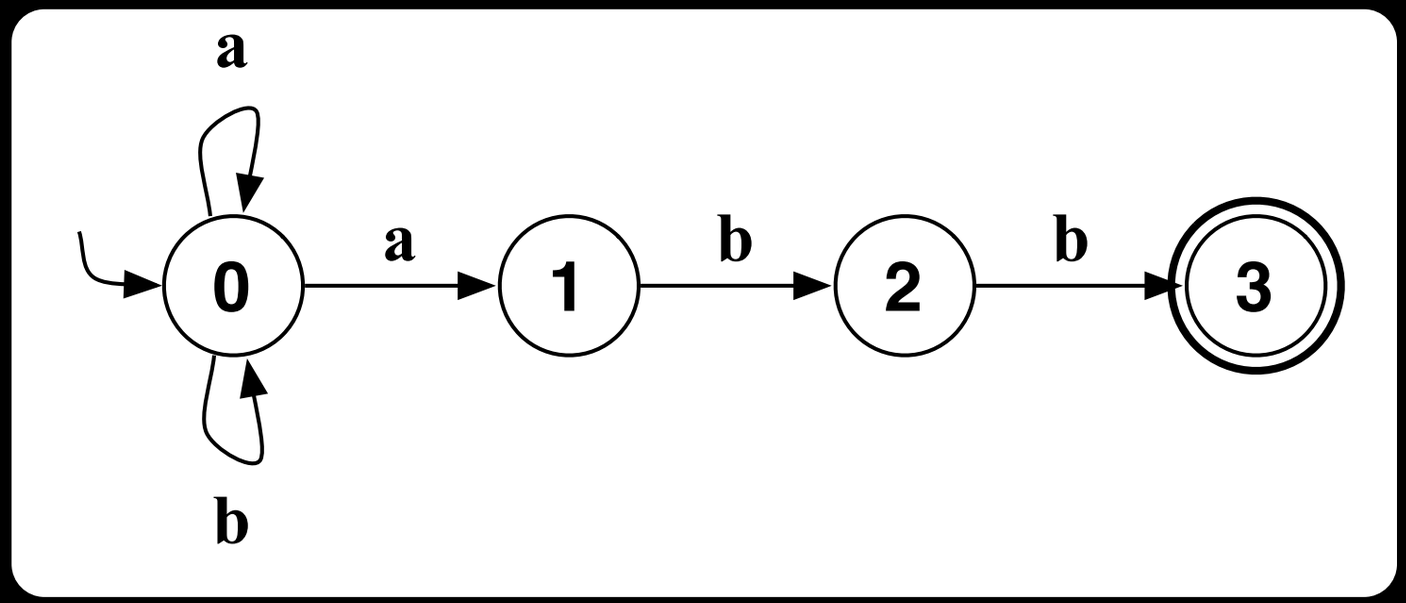
- *stmt* → **if** *expr* **then** *stmt* **else** *stmt*
- Um conjunto de símbolos terminais (*tokens*).
- Um conjunto de símbolos não-terminais.
- Um conjunto de produções, cada produção consiste de um não-terminal, uma seta, e uma seqüencia de *tokens* e/ou não terminais
- Um não terminal designado como símbolo inicial

Expressões Regulares ***vs.*** Gramáticas Livres de Contexto

- Tudo que pode ser descrito por uma ER pode ser descrito com GLC, mas:
  - Regras léxicas são especificadas mais simplesmente com ER
  - ER geralmente são mais concisas e simples
  - Podem ser gerados analisadores léxicos mais eficientes a partir de expressões regulares
  - Estrutura/modulariza o front-end do compilador

(a | b)*abb

$$
\begin{aligned}
A_0 &\rightarrow \mathbf{a}A_0 \mid \mathbf{b}A_0 \mid \mathbf{a}A_1 \\
A_1 &\rightarrow \mathbf{b}A_2 \\
A_2 &\rightarrow \mathbf{b}A_3 \\
A_3 &\rightarrow \varepsilon
\end{aligned}
$$



Expressões Regulares ***vs.*** Gramáticas Livres de Contexto

- Expressões regulares são convenientes para especificar a estrutura de construções léxicas, como identificadores, constantes, palavras chave etc.

- Usamos gramáticas para especificar estruturas aninhadas, como parênteses, begin-end, if-then-else etc.

Limites de especificação com gramáticas livres de contexto

- L = {wcw | w está em (a|b)*}
  ex: declaração de variáveis antes de seu uso

- L = {$a^n b^m c^n d^m$ | n >= 1 e m >= 1}
  ex: contagem do número de argumentos de funções/procedimentos.

Que palavras pertencem a L(G)?

- aba
- acca
- abcba
- abcbcba

$$S \rightarrow \mathrm{a}X\mathrm{a}$$
$$X \rightarrow \mathrm{b}Y \mid \varepsilon$$
$$Y \rightarrow \mathrm{c}X\mathrm{c} \mid \varepsilon$$

Derivação

- Derivação: Dada uma gramática G, produz uma string s que faz parte de L(G)

Gramática G

$$S \rightarrow \mathbf{a}AB\mathbf{e}$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$

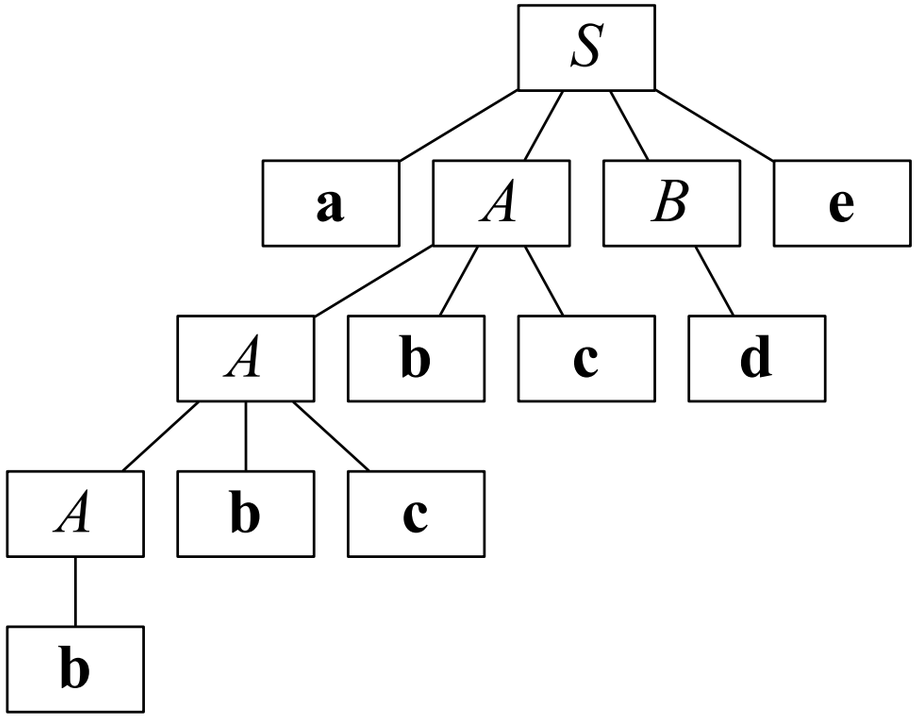
Derivação de abbcbcde

$$\begin{aligned}
&S \\
&\rightarrow \mathbf{a}\underline{A}B\mathbf{e} \\
&\rightarrow \mathbf{a}\underline{A}\mathbf{bc}B\mathbf{e} \\
&\rightarrow \mathbf{a}\underline{A}\mathbf{bcbc}B\mathbf{e} \\
&\rightarrow \mathbf{abbcbc}\underline{B}\mathbf{e} \\
&\rightarrow \mathbf{abbcbcde}
\end{aligned}$$

Parsing

- Dada uma string *s* em L(G), produz uma árvore sintática que demonstra como obter a derivação de *s*

Gramática G



Derivação de abbcbcde

$$
\begin{aligned}
&S \\
&\rightarrow \mathbf{a}\underline{A}B\mathbf{e} \\
&\rightarrow \mathbf{a}\underline{A}\mathbf{bc}B\mathbf{e} \\
&\rightarrow \mathbf{a}\underline{A}\mathbf{bcbc}B\mathbf{e} \\
&\rightarrow \mathbf{abbcbc}\underline{B}\mathbf{e} \\
&\rightarrow \mathbf{abbcbcde}
\end{aligned}
$$

Categorias

- Métodos de parsing universais: funcionam para qualquer gramática, mas são muito ineficientes (inviáveis para uso prático).

- Top down: constroem as *parse trees* a partir da raiz em direção às folhas.

- Bottom-up: constroem as *parse trees* a partir das folhas.

Top-Down Parsing

O Método

- A partir do símbolo inicial da gramática, consuma tokens da esquerda para direita
- Decida que produção aplicar, de acordo com o token retornado
- Continue até que
  - todas as folhas sejam símbolos terminais e não há mais tokens a serem lidos da entrada
  - há uma falta de correspondência entre a entrada e as folhas da *parse tree* parcialmente construída

No primeiro caso,
tudo ok!

E no segundo?

Exemplo

$$S \rightarrow \mathbf{a}AB\mathbf{e}$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	S

Exemplo

$S \rightarrow \mathbf{a}AB\mathbf{e}$

$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	S

Exemplo

$S \rightarrow \mathbf{a}AB\mathbf{e}$

$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

a	b	b	c	b	c	d	e	a*ABe*

Exemplo

$S \rightarrow \mathbf{a}AB\mathbf{e}$

$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	$AB\mathbf{e}$

Exemplo

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow Abc \mid b$

$B \rightarrow \mathbf{d}$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	ABe

escolher uma produção!

Exemplo

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	ABe

Exemplo

$S \rightarrow \mathbf{a}AB\mathbf{e}$

$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	$\mathbf{b}B\mathbf{e}$

Exemplo

$$S \rightarrow \mathbf{a}AB\mathbf{e}$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	$B\mathbf{e}$

Erro! Backtrack...

Exemplo

restaura estado anterior à última escolha

$S \rightarrow \mathbf{a}AB\mathbf{e}$

$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	$AB\mathbf{e}$

faz outra escolha!

Exemplo

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow Abc \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	ABe

Exemplo

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow Abc \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	$A\mathbf{bc}B\mathbf{e}$

Exemplo

$$S \rightarrow \mathbf{a}ABe$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$

Símbolo A novamente na produção mais à esquerda

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	$A\mathbf{bc}B\mathbf{e}$

De novo!
Escolher uma produção!

Exemplo

$S \rightarrow \mathbf{a}AB\mathbf{e}$

$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	$A\mathbf{bc}B\mathbf{e}$

Exemplo

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow Abc \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	**bbc***B***e**

Exemplo

$$S \rightarrow \mathbf{a}AB\mathbf{e}$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	**bc**B**e**

Exemplo

$$S \rightarrow \mathbf{a}AB\mathbf{e}$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	$\mathbf{c}B\mathbf{e}$

Exemplo

$$S \rightarrow \mathbf{a}AB\mathbf{e}$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$

a	b	b	c	b	c	d	e	Be

Erro! Backtrack...

Exemplo

restaura estado anterior à última escolha

$$S \rightarrow \mathbf{a}AB\mathbf{e}$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	$A\mathbf{bc}B\mathbf{e}$

faz outra escolha!

Exemplo

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow Abc \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	$A\mathbf{bc}B\mathbf{e}$

Exemplo

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow Abc \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	$A\mathbf{bcbc}B\mathbf{e}$

Exemplo

$$S \rightarrow \mathbf{a}AB\mathbf{e}$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$

Símbolo A novamente na produção mais à esquerda

a	b	b	c	b	c	d	e	$A\mathbf{bcbc}B\mathbf{e}$

De novo!
Escolher uma produção!

Exemplo

$$S \rightarrow \mathbf{a}ABe$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$

a	b	b	c	b	c	d	e	Abcbc$B$e

Exemplo

$$S \rightarrow \mathbf{a}ABe$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$

a	b	b	c	b	c	d	e	bbcbc*B*e

Exemplo

$$S \rightarrow \mathbf{a}ABe$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	**bcbc**B**e**

Exemplo

$$S \rightarrow \mathbf{a}AB\mathbf{e}$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	**cbc***B***e**

Exemplo

$S \rightarrow \mathbf{a}AB\mathbf{e}$

$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	**bc***B***e**

Exemplo

$$S \rightarrow \mathbf{a}AB\mathbf{e}$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	**c***B***e**

Exemplo

$S \rightarrow \mathbf{a}AB\mathbf{e}$

$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	$B\mathbf{e}$

Exemplo

$$S \rightarrow \mathbf{a}ABe$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$

Símbolo B na produção mais à esquerda

a	b	b	c	b	c	d	e	Be

Só tem uma escolha…

Exemplo

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	Be

Exemplo

$$S \rightarrow \mathbf{a}ABe$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$

a b b c b c d e | **de**

Exemplo

$$S \rightarrow \mathbf{a}AB\mathbf{e}$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$

a	**b**	**b**	**c**	**b**	**c**	**d**	**e**	**e**

Exemplo

$S \rightarrow \mathbf{a}AB\mathbf{e}$

$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

a	b	b	c	b	c	d	e	$

Fim do arquivo

- Parsers devem ler não apenas os símbolos terminais, mas também o marcador de fim de arquivo
- Usamos $ para representar o fim do arquivo
  - $S' \rightarrow S\$$

Exemplo

$$
\begin{aligned}
S &\rightarrow \mathbf{a}A B \mathbf{e} \\
A &\rightarrow A\mathbf{bc} \mid \mathbf{b} \\
B &\rightarrow \mathbf{d}
\end{aligned}
$$

a	b	b	c	b	c	d	e	$

Derivação correspondente!

$$
\begin{aligned}
&S \\
&\rightarrow \mathbf{a}\underline{A}B\mathbf{e} \\
&\rightarrow \mathbf{a}\underline{A}\mathbf{bc}B\mathbf{e} \\
&\rightarrow \mathbf{a}\underline{A}\mathbf{bcbc}B\mathbf{e} \\
&\rightarrow \mathbf{abbcbc}\underline{B}\mathbf{e} \\
&\rightarrow \mathbf{abbcbcde}
\end{aligned}
$$

Backtracking: A maior fonte de ineficiência em um *parser top-down* e *leftmost*

Seria interessante termos *parsers* que não fazem *backtracking*

predictive parsing

O que é importante na gramática para que o parser seja preditivo?

O token lido como primeiro terminal deve fornecer informação suficiente para decidirmos que produção aplicar

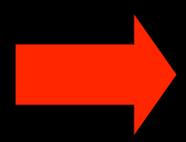
Que situações normalmente geram problemas para *predictive parsing*?

Recursão à esquerda

- Uma gramática é recursiva à esquerda se existe um não terminal A tal que existe uma derivação de A que gera $A\alpha$, para alguma string α.

- Existem técnicas para eliminar a recursão à esquerda, já vistas nas aulas anteriores.

$$S \rightarrow \mathbf{a}AB\mathbf{e}$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$



$$S \rightarrow \mathbf{a}AB\mathbf{e}$$
$$A \rightarrow \boldsymbol{b}K$$
$$K \rightarrow \boldsymbol{bc}K \mid \varepsilon$$
$$B \rightarrow \mathbf{d}$$

Eliminando recursão à esquerda

$$
\begin{aligned}
S &\rightarrow A\mathbf{a} \mid \mathbf{b} \\
A &\rightarrow A\mathbf{c} \mid S\mathbf{d} \mid \boldsymbol{\varepsilon}
\end{aligned}
$$

Eliminando recursão à esquerda

$$
\begin{aligned}
S &\rightarrow A\mathbf{a} \mid \mathbf{b} \\
A &\rightarrow A\mathbf{c} \mid A\mathbf{ad} \mid \mathbf{bd} \mid \varepsilon
\end{aligned}
$$

Solução

S → A**a** | **b**

A → **bd**A’ | A’

A’ → **c**A’ | **ad**A’ | **ε**

Fatoração à Esquerda

- Técnica de transformação de gramática usada para produzir uma gramática adequada para predictive parsing.

- Combina os casos em que há mais de uma alternativa possível a partir do reconhecimento de um token.

- Existem algoritmos para fazer a fatoração à esquerda.

Fatoração à Esquerda - exemplo

stmt → **if** *expr* **then** *stmt*

stmt → **if** *expr* **then** *stmt* **else** *stmt*

stmt → **other**

Fatoração à Esquerda - exemplo

$S \rightarrow \mathbf{i}\,E\,\mathbf{t}\,S \mid \mathbf{i}\,E\,\mathbf{t}\,S\,\mathbf{e}\,S \mid \mathbf{a}$

$E \rightarrow \mathbf{b}$

Fatoração à Esquerda - solução

$S \rightarrow \mathbf{i}\,E\,\mathbf{t}\,S \mid \mathbf{i}\,E\,\mathbf{t}\,S\,\mathbf{e}\,S \mid \mathbf{a}$

$E \rightarrow \mathbf{b}$

$S \rightarrow \mathbf{i}\,E\,\mathbf{t}\,S\,S' \mid \mathbf{a}$

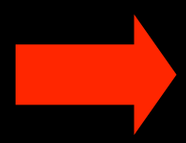
$S' \rightarrow \mathbf{e}\,S \mid \boldsymbol{\varepsilon}$

$E \rightarrow \mathbf{b}$

Predictive parsing

- Na gramática original, A não tem informação suficiente para decidir qual produção aplicar ao enxergar um $\mathbf{b}$ no lookahead
- Na gramática modificada, cada produção alternativa do mesmo não terminal inicia com terminais distintos

$$S \rightarrow \mathbf{a}ABe$$
$$A \rightarrow A\mathbf{bc} \mid \mathbf{b}$$
$$B \rightarrow \mathbf{d}$$



$$S \rightarrow \mathbf{a}ABe$$
$$A \rightarrow \boldsymbol{b}K$$
$$K \rightarrow \boldsymbol{bc}K \mid \varepsilon$$
$$B \rightarrow \mathbf{d}$$

Classificação de Parsers

Número de *tokens* de *lookahead* lidos, porém, não consumidos

Lê entrada da esquerda (L) para direita

LL(k)

Procura derivação mais à esquerda

FIRST e FOLLOW

- A construção de parsers top-down e bottom-up é auxiliada por duas funções, FIRST e FOLLOW

- Estas funções auxiliam o parser a decidir qual produção será aplicada, baseado no próximo símbolo de entrada

- Em condições de recuperação de erro, também podem ser úteis

FIRST

- $\mathrm{FIRST}(\alpha)$, onde α é qualquer string de símbolos da gramática, é o conjunto de terminais que iniciam strings derivadas a partir de α
- Se α pode gerar ε, ε pertence a $\mathrm{FIRST}(\alpha)$
- Considere duas produções $A \rightarrow \alpha \mid \beta$
  - Para que o parser seja preditivo, o que deve ser verdade?

FIRST

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow \mathbf{b}K$

$K \rightarrow \mathbf{bc}K$

$K \rightarrow \varepsilon$

$B \rightarrow \mathbf{d}$

	FIRST	FOLLOW
S		
A		
K		
B		

FIRST

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow \mathbf{b}K$

$K \rightarrow \mathbf{bc}K$

$K \rightarrow \varepsilon$

$B \rightarrow \mathbf{d}$

	FIRST	FOLLOW
S	{ **a** }	
A		
K		
B		

FIRST

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow \mathbf{b}K$

$K \rightarrow \mathbf{bc}K$

$K \rightarrow \varepsilon$

$B \rightarrow \mathbf{d}$

	FIRST	FOLLOW
S	{ **a** }	
A	{ **b** }	
K		
B		

FIRST

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow \mathbf{b}K$

$K \rightarrow \mathbf{bc}K$

$K \rightarrow \varepsilon$

$B \rightarrow \mathbf{d}$

	FIRST	FOLLOW
S	{ **a** }	
A	{ **b** }	
K	{ **b**, **ε** }	
B		

FIRST

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow \mathbf{b}K$

$K \rightarrow \mathbf{bc}K$

$K \rightarrow \varepsilon$

$B \rightarrow \mathbf{d}$

	FIRST	FOLLOW
S	{ **a** }	
A	{ **b** }	
K	{ **b**, **ε** }	
B	{ **d** }	

Calculando $\mathrm{FIRST}(X)$

- Se X é um terminal $\mathrm{FIRST}(X) = \{X\}$;
- Se $X \rightarrow \varepsilon$, $\varepsilon \in \mathrm{FIRST}(X)$
- Se $X \rightarrow Y_1 Y_2 \ldots Y_k$, $\mathrm{FIRST}(Y_1 Y_2 \ldots Y_k) \subseteq \mathrm{FIRST}(X)$
  - $\mathrm{FIRST}(Y_1 Y_2 \ldots Y_k)$ é
    - $[\varepsilon \notin \mathrm{FIRST}(Y_1)]$ $\mathrm{FIRST}(Y_1)$**; ou**
    - $[\varepsilon \in \mathrm{FIRST}(Y_1)]$ $\mathrm{FIRST}(Y_1)$-$\{\varepsilon\} \cup \mathrm{FIRST}(Y_2 \ldots Y_k)$
    - Se $\varepsilon \in \mathrm{FIRST}(Y_j)$ para todo j de 1 a k, $\varepsilon \in \mathrm{FIRST}(Y_1 Y_2 \ldots Y_k)$

FOLLOW

- FOLLOW(A), onde A é um não-terminal, é o conjunto de terminais α que pode aparecer imediatamente à direita de A

- Se A pode ser a última produção à direita, $ pertence a FOLLOW(A)

FOLLOW

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow \mathbf{b}K$

$K \rightarrow \mathbf{bc}K$

$K \rightarrow \varepsilon$

$B \rightarrow \mathbf{d}$

	FIRST	FOLLOW
S	**{ a }**	
A	**{ b }**	
K	**{ b, ε }**	
B	**{ d }**	

FOLLOW

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow \mathbf{b}K$

$K \rightarrow \mathbf{bc}K$

$K \rightarrow \varepsilon$

$B \rightarrow \mathbf{d}$

	FIRST	FOLLOW
S	{ **a** }	{ **$** }
A	{ **b** }	
K	{ **b**, **ε** }	
B	{ **d** }	

FOLLOW

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow \mathbf{b}K$

$K \rightarrow \mathbf{bc}K$

$K \rightarrow \varepsilon$

$B \rightarrow \mathbf{d}$

	FIRST	FOLLOW
S	{ **a** }	{ **$** }
A	{ **b** }	{ **d** }
K	{ **b**, **ε** }	
B	{ **d** }	

FOLLOW

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow \mathbf{b}K$

$K \rightarrow \mathbf{bc}K$

$K \rightarrow \varepsilon$

$B \rightarrow \mathbf{d}$

	FIRST	FOLLOW
S	{ **a** }	{ **$** }
A	{ **b** }	{ **d** }
K	{ **b**, **ε** }	{ **d** }
B	{ **d** }	

FOLLOW

$S \rightarrow \mathbf{a}ABe$

$A \rightarrow \mathbf{b}K$

$K \rightarrow \mathbf{bc}K$

$K \rightarrow \varepsilon$

$B \rightarrow \mathbf{d}$

	FIRST	FOLLOW
S	**{ a }**	**{ $ }**
A	**{ b }**	**{ d }**
K	**{ b, ε }**	**{ d }**
B	**{ d }**	**{ e }**

Calculando $\mathbf{FOLLOW}$

- $\$ \in \mathbf{FOLLOW}(S)$, onde S é o símbolo inicial e $ é fim da entrada
- Se existe uma produção $A \rightarrow \alpha B \beta$, tudo que pertence a $\mathbf{FIRST}(\beta)$ exceto ε está em $\mathbf{FOLLOW}(B)$
- Se existe uma produção $A \rightarrow \alpha B$, então tudo que estiver em $\mathbf{FOLLOW}(A)$ estará em $\mathbf{FOLLOW}(B)$
- Se existe uma produção $A \rightarrow \alpha B \beta$, e $\varepsilon \in \mathbf{FIRST}(\beta)$, tudo que estiver em $\mathbf{FOLLOW}(A)$ estará em $\mathbf{FOLLOW}(B)$