Linguagens e Programação

Gramáticas

Ana Madureira
amd@isep.ipp.pt

António Silva
ass@isep.ipp.pt

Paulo Ferreira
pdf@isep.ipp.pt

Bruno Cunha
cun@isep.ipp.pt

Instituto Superior de Engenharia do Porto

2021/2022

O que é uma gramática?

- Uma gramática é uma ferramenta poderosa para a descrição e análise de linguagens;
- Constituída por um conjunto de regras segundo as quais as frases válidas da linguagem são construídas;
- Constituída por vários tipos de palavras (*tokens*) normalmente reconhecidos pelo **analisador léxico**;

Definição formal

- Formalmente, uma gramática é definida pelo tuplo $G = (V, \Sigma, P, S)$, no qual:

Definição formal

- Formalmente, uma gramática é definida pelo tuplo $G = (V, \Sigma, P, S)$, no qual:
  - V – um conjunto finito, não vazio, de **variáveis** (símbolos **não terminais**);

Definição formal

- Formalmente, uma gramática é definida pelo tuplo $G = (V, \Sigma, P, S)$, no qual:
  - V – um conjunto finito, não vazio, de **variáveis** (símbolos **não terminais**);
  - Σ – é um conjunto finito, não vazio, dito **alfabeto** ou conjunto de símbolos **terminais**;

Definição formal

- Formalmente, uma gramática é definida pelo tuplo $G = (V, \Sigma, P, S)$, no qual:
  - V – um conjunto finito, não vazio, de **variáveis** (símbolos **não terminais**);
  - Σ – é um conjunto finito, não vazio, dito **alfabeto** ou conjunto de símbolos **terminais**;
  - P – conjunto de **produções**, regras da gramática. A sua forma geral é a seguinte: $\alpha \rightarrow \beta$ que definem a forma como o conjunto de símbolos α podem ser substituídos pelo conjunto de símbolos β;

Definição formal

- Formalmente, uma gramática é definida pelo tuplo $G = (V, \Sigma, P, S)$, no qual:
  - V – um conjunto finito, não vazio, de **variáveis** (símbolos **não terminais**);
  - Σ – é um conjunto finito, não vazio, dito **alfabeto** ou conjunto de símbolos **terminais**;
  - P – conjunto de **produções**, regras da gramática. A sua forma geral é a seguinte: $\alpha \rightarrow \beta$ que definem a forma como o conjunto de símbolos α podem ser substituídos pelo conjunto de símbolos β;
  - S – **símbolo inicial** a partir do qual todas as frases são derivadas.

Notação BNF

A notação BNF (*Backus Naur Form* ou *Backus Normal Form*) foi originalmente criada por **John Backus** e **Peter Naur**, no final dos anos 50, para descrever a linguagem ALGOL.

Os meta-símbolos utilizados na notação BNF são:

::= — "definido como";
| — alternativa;
< > — regra

Notação BNF

A notação BNF (*Backus Naur Form* ou *Backus Normal Form*) foi originalmente criada por **John Backus** e **Peter Naur**, no final dos anos 50, para descrever a linguagem ALGOL.

Os meta-símbolos utilizados na notação BNF são:

::= — “definido como”;

| — alternativa;

$< \, >$ — regra

$$<\textbf{numero}> ::= \ <\textbf{algarismo}> | <\textbf{algarismo}><\textbf{numero}>$$

$$<\textbf{algarismo}> ::= \ \mathit{0} \,|\, \mathit{1} \,|\, \mathit{2} \,|\, \ldots \,|\, \mathit{8} \,|\, \mathit{9}$$

Notação EBNF

A notação EBNF extende a notação BNF com os seguinte meta-símbolos:

- [ ] — parte opcional;
- { } — parte que se pode repetir 0 ou mais vezes;
- ( ) — precedências dentro da regra ;
- “ ” — carácter a tratar como símbolo terminal *e.g.*, “<”.

Notação EBNF

A notação EBNF extende a notação BNF com os seguinte meta-símbolos:

[ ] — parte opcional;
{ } — parte que se pode repetir 0 ou mais vezes;
( ) — precedências dentro da regra ;
“ ” — carácter a tratar como símbolo terminal *e.g.*, “$<$”.

$$<\textbf{identificador}> ::= (<\textbf{letra}> \mid \_)\{(<\textbf{letra}> \mid <\textbf{algarismo}> \mid \_)\}$$

$$<\textbf{letra}> ::= a \mid A \mid b \mid ... \mid z \mid Z$$

$$<\textbf{algarismo}> ::= 0 \mid 1 \mid 2 \mid ... \mid 8 \mid 9$$

Exemplo de uma gramática

$$
\begin{aligned}
<\textbf{frase}> &\rightarrow <\textbf{sintagma-nominal}><\textbf{sintagma-verbal}> \\
<\textbf{sintagma-nominal}> &\rightarrow <\textbf{artigo}><\textbf{nome}> \mid <\textbf{nome}> \\
<\textbf{sintagma-verbal}> &\rightarrow <\textbf{verbo}><\textbf{sintagma-nominal}> \\
<\textbf{artigo}> &\rightarrow \textit{o} \mid \textit{a} \mid \textit{os} \mid \textit{as} \\
<\textbf{nome}> &\rightarrow \textit{Pedro} \mid \textit{Maria} \mid \textit{crianças} \mid \textit{rapazes} \mid \textit{cartas} \mid \textit{futebol} \\
<\textbf{verbo}> &\rightarrow \textit{conhece} \mid \textit{conhecem} \mid \textit{é} \mid \textit{são} \mid \textit{joga} \mid \textit{jogam}
\end{aligned}
$$

Exemplo de uma gramática

$$
\begin{aligned}
<\textbf{frase}> &\rightarrow <\textbf{sintagma-nominal}><\textbf{sintagma-verbal}>\\
<\textbf{sintagma-nominal}> &\rightarrow <\textbf{artigo}><\textbf{nome}> \mid <\textbf{nome}>\\
<\textbf{sintagma-verbal}> &\rightarrow <\textbf{verbo}><\textbf{sintagma-nominal}>\\
<\textbf{artigo}> &\rightarrow \textit{o} \mid \textit{a} \mid \textit{os} \mid \textit{as}\\
<\textbf{nome}> &\rightarrow \textit{Pedro} \mid \textit{Maria} \mid \textit{crianças} \mid \textit{rapazes} \mid \textit{cartas} \mid \textit{futebol}\\
<\textbf{verbo}> &\rightarrow \textit{conhece} \mid \textit{conhecem} \mid \textit{é} \mid \textit{são} \mid \textit{joga} \mid \textit{jogam}
\end{aligned}
$$

Com estas regras (também designadas por produções) é possível analisar frases como as seguintes:

os rapazes jogam futebol
o Pedro conhece a Maria
a Maria é criança

Exemplo de uma gramática

$$
\begin{aligned}
<\textbf{frase}> &\rightarrow <\textbf{sintagma-nominal}><\textbf{sintagma-verbal}> \\
<\textbf{sintagma-nominal}> &\rightarrow <\textbf{artigo}><\textbf{nome}> \mid <\textbf{nome}> \\
<\textbf{sintagma-verbal}> &\rightarrow <\textbf{verbo}><\textbf{sintagma-nominal}> \\
<\textbf{artigo}> &\rightarrow \textit{o} \mid \textit{a} \mid \textit{os} \mid \textit{as} \\
<\textbf{nome}> &\rightarrow \textit{Pedro} \mid \textit{Maria} \mid \textit{crianças} \mid \textit{rapazes} \mid \textit{cartas} \mid \textit{futebol} \\
<\textbf{verbo}> &\rightarrow \textit{conhece} \mid \textit{conhecem} \mid \textit{é} \mid \textit{são} \mid \textit{joga} \mid \textit{jogam}
\end{aligned}
$$

Derivação de frases

$u \Rightarrow v$ — diz-se que u deriva em v num passo

$u \Rightarrow^{*} v$ — diz-se que u deriva em v em zero ou mais passos

$u \Rightarrow^{+} v$ — diz-se que u deriva em v em um ou mais passos

Exemplo de uma gramática

$$
\begin{aligned}
<\textbf{frase}> &\rightarrow <\textbf{sintagma-nominal}><\textbf{sintagma-verbal}>\\
<\textbf{sintagma-nominal}> &\rightarrow <\textbf{artigo}><\textbf{nome}> \mid <\textbf{nome}>\\
<\textbf{sintagma-verbal}> &\rightarrow <\textbf{verbo}><\textbf{sintagma-nominal}>\\
<\textbf{artigo}> &\rightarrow \textit{o} \mid \textit{a} \mid \textit{os} \mid \textit{as}\\
<\textbf{nome}> &\rightarrow \textit{Pedro} \mid \textit{Maria} \mid \textit{crianças} \mid \textit{rapazes} \mid \textit{cartas} \mid \textit{futebol}\\
<\textbf{verbo}> &\rightarrow \textit{conhece} \mid \textit{conhecem} \mid \textit{é} \mid \textit{são} \mid \textit{joga} \mid \textit{jogam}
\end{aligned}
$$

A linguagem gerada por uma gramática $L(G)$ é dada pelo conjunto de todas as derivações que, partindo do estado inicial S, originam uma sequência de símbolos do alfabeto:

$$L(G) = \{u \in \Sigma^* : S \Rightarrow^* u\}$$

Derivação de frases

$$
\begin{aligned}
<\textbf{frase}> &\rightarrow <\textbf{sintagma-nominal}><\textbf{sintagma-verbal}> \\
<\textbf{sintagma-nominal}> &\rightarrow <\textbf{artigo}><\textbf{nome}> \mid <\textbf{nome}> \\
<\textbf{sintagma-verbal}> &\rightarrow <\textbf{verbo}><\textbf{sintagma-nominal}> \\
<\textbf{artigo}> &\rightarrow \textit{o} \mid \textit{a} \mid \textit{os} \mid \textit{as} \\
<\textbf{nome}> &\rightarrow \textit{Pedro} \mid \textit{Maria} \mid \textit{crianças} \mid \textit{rapazes} \mid \textit{cartas} \mid \textit{futebol} \\
<\textbf{verbo}> &\rightarrow \textit{conhece} \mid \textit{conhecem} \mid \textit{é} \mid \textit{são} \mid \textit{joga} \mid \textit{jogam}
\end{aligned}
$$

$$
\begin{aligned}
<\textbf{frase}> &\Rightarrow <\textbf{sintagma-nominal}><\textbf{sintagma-verbal}> \\
&\Rightarrow <\textbf{artigo}><\textbf{nome}><\textbf{sintagma-verbal}> \\
&\Rightarrow \textit{os}\ <\textbf{nome}><\textbf{sintagma-verbal}> \\
&\Rightarrow \textit{os}\ \textit{rapazes}\ <\textbf{sintagma-verbal}> \\
&\Rightarrow \textit{os}\ \textit{rapazes}\ <\textbf{verbo}><\textbf{sintagma-nominal}>
\end{aligned}
$$

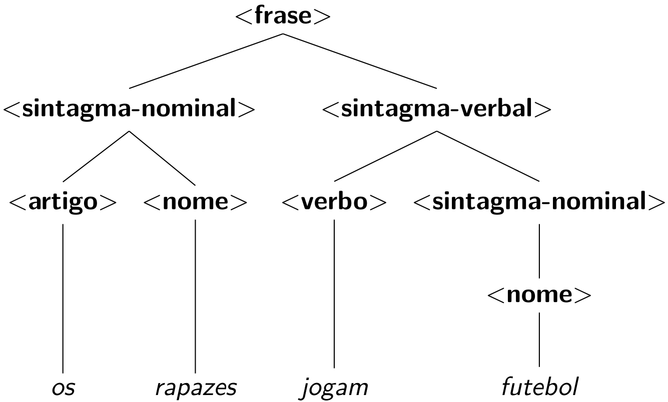
Derivação de frases

$$
\begin{aligned}
<\textbf{frase}> &\rightarrow <\textbf{sintagma-nominal}><\textbf{sintagma-verbal}> \\
<\textbf{sintagma-nominal}> &\rightarrow <\textbf{artigo}><\textbf{nome}> \mid <\textbf{nome}> \\
<\textbf{sintagma-verbal}> &\rightarrow <\textbf{verbo}><\textbf{sintagma-nominal}> \\
<\textbf{artigo}> &\rightarrow \textit{o} \mid \textit{a} \mid \textit{os} \mid \textit{as} \\
<\textbf{nome}> &\rightarrow \textit{Pedro} \mid \textit{Maria} \mid \textit{crianças} \mid \textit{rapazes} \mid \textit{cartas} \mid \textit{futebol} \\
<\textbf{verbo}> &\rightarrow \textit{conhece} \mid \textit{conhecem} \mid \textit{é} \mid \textit{são} \mid \textit{joga} \mid \textit{jogam}
\end{aligned}
$$

$$
\begin{aligned}
<\textbf{frase}> &\Rightarrow^{*} \textit{os rapazes} <\textbf{verbo}><\textbf{sintagma-nominal}> \\
&\Rightarrow \textit{os rapazes jogam} <\textbf{sintagma-nominal}> \\
&\Rightarrow \textit{os rapazes jogam} <\textbf{nome}> \\
&\Rightarrow \textit{os rapazes jogam futebol}
\end{aligned}
$$

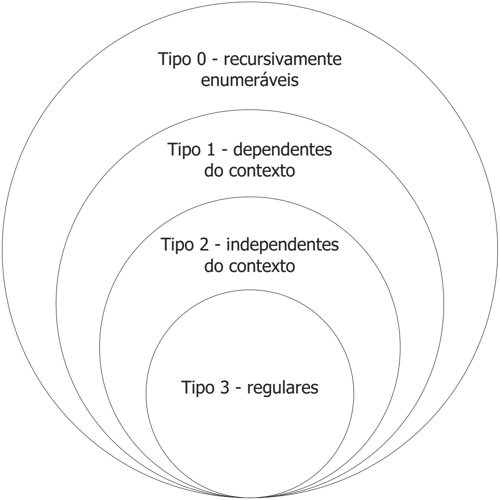
Árvores de derivação

O processo de derivação pode ser representado graficamente através de uma árvore de derivação



Hierarquia de Chomsky

Noam Chomsky (linguista americano) criou uma hierarquia para as gramáticas formais, dividindo-as em quatro tipos



Gramáticas do tipo 3

Gramáticas regulares. A produções são da forma:

$$A \to a$$
$$A \to aB$$
$$A \to \varepsilon$$

em que $\boldsymbol{A}$ e $\boldsymbol{B}$ são dois quaisquer **não terminais** singulares e $\boldsymbol{a}$ é um qualquer **terminal** singular.

Estas são as formas de gramáticas mais restritas em termos de poder de representação.

Nota: A produção $A \to a$ pode ser omitida, pois pode ser derivada a partir das outras duas.

Gramáticas do tipo 3

Gramáticas regulares. A produções são da forma:

$$A \rightarrow a$$
$$A \rightarrow aB$$
$$A \rightarrow \varepsilon$$

em que $\boldsymbol{A}$ e $\boldsymbol{B}$ são dois quaisquer **não terminais** singulares e $\boldsymbol{a}$ é um qualquer **terminal** singular.

Estas são as formas de gramáticas mais restritas em termos de poder de representação.

Lineares à direita

$$A \rightarrow a$$
$$A \rightarrow aB$$
$$A \rightarrow \varepsilon$$

Lineares à esquerda

$$A \rightarrow a$$
$$A \rightarrow Ba$$
$$A \rightarrow \varepsilon$$

Gramáticas do tipo 2

Gramáticas independentes do contexto. A produções são da forma:

$$A \to \alpha$$

em que $\boldsymbol{\alpha}$ é uma sequência arbitrária de símbolos terminais e não terminais, sendo $\boldsymbol{A}$ um qualquer não terminal singular.

Tal significa que qualquer ocorrência de $\boldsymbol{A}$ pode ser substituída por $\boldsymbol{\alpha}$ independentemente do contexto.

Gramáticas do tipo 2

Gramáticas independentes do contexto. A produções são da forma:

$$A \rightarrow \alpha$$

em que $\boldsymbol{\alpha}$ é uma sequência arbitrária de símbolos terminais e não terminais, sendo $\boldsymbol{A}$ um qualquer não terminal singular.

Tal significa que qualquer ocorrência de $\boldsymbol{A}$ pode ser substituída por $\boldsymbol{\alpha}$ independentemente do contexto.

Reconhece frases do tipo $\{a^n b^p d^q c^n : n, p, q \geq 0\}$

Exemplo

$$A \rightarrow aAc$$
$$A \rightarrow BD$$
$$B \rightarrow bB|\varepsilon$$
$$D \rightarrow dD|\varepsilon$$

Gramáticas do tipo 1

Gramáticas dependentes do contexto. A produções são da forma:

$$\alpha A\beta \to \alpha\gamma\beta$$

em que $\boldsymbol{\alpha}$,$\boldsymbol{\beta}$ e $\boldsymbol{\gamma}$ são sequências arbitrárias de símbolos terminais e não terminais, sendo que γ não é nulo e $\boldsymbol{A}$ é um qualquer não terminal singular.

Estas gramáticas têm que respeitar a condição $|\boldsymbol{\alpha A\beta}| \leq |\boldsymbol{\alpha\gamma\beta}|$

A única excepção à regra anterior, é a produção inicial ser $S \to \varepsilon$, mas **se e só se** o S não existir do lado direito (para permitir a palavra vazia).

Gramáticas do tipo 1

Gramáticas dependentes do contexto. A produções são da forma:

$$\alpha A\beta \rightarrow \alpha\gamma\beta$$

em que $\boldsymbol{\alpha}$,$\boldsymbol{\beta}$ e $\boldsymbol{\gamma}$ são sequências arbitrárias de símbolos terminais e não terminais, sendo que $\boldsymbol{\gamma}$ não é nulo e $\boldsymbol{A}$ é um qualquer não terminal singular.

Reconhece frases do tipo
$\{a^n b^n c^n : n \geq 1\}$

Exemplo

$$S \rightarrow aSBC|abC$$
$$CB \rightarrow XB$$
$$XB \rightarrow XC$$
$$XC \rightarrow BC$$
$$bB \rightarrow bb$$
$$bC \rightarrow bc$$
$$cC \rightarrow cc$$

Gramáticas do tipo 0

Gramáticas livres ou sem restrições. A produções são da forma:

$$\alpha \to \beta$$

em que tanto $\boldsymbol{\alpha}$ como $\boldsymbol{\beta}$ são sequências arbitrárias de símbolos terminais e não terminais.

O lado esquerdo da produção não pode ser vazio.

Gramáticas do tipo 0

Gramáticas livres ou sem restrições. A produções são da forma:

$$\alpha \to \beta$$

em que tanto α como β são sequências arbitrárias de símbolos terminais e não terminais.

O lado esquerdo da produção não pode ser vazio.

Reconhece frases do tipo $\{a^n b^n c^n : n \geq 1\}$

Exemplo

$$S \to aSBC | abC$$
$$CB \to BC$$
$$bB \to bb$$
$$bC \to bc$$
$$cC \to cc$$

Notação BNF (Exemplo avançado)

A notação BNF (*Backus Naur Form* ou *Backus Normal Form*) foi originalmente criada por **John Backus** e **Peter Naur**, no final dos anos 50, para descrever a linguagem ALGOL.

$$
\begin{aligned}
&<\textbf{if}> ::= \textit{if}<\textbf{condição}>\ \textit{then}\ <\textbf{instruções}>\ \textit{endif}\\
&\qquad\qquad \mid \textit{if}\ <\textbf{condição}>\ \textit{then}\ <\textbf{instruções}>\ \textit{else}\ <\textbf{instruções}>\ \textit{endif}\\
&<\textbf{identificador}> ::= <\textbf{letra}><\textbf{alfanums}> \mid \_<\textbf{alfanums}>\\
&<\textbf{alfanums}> ::= <\textbf{alfanum}><\textbf{alfanums}> \mid \varepsilon\\
&<\textbf{alfanum}> ::= <\textbf{letra}> \mid <\textbf{algarismo}> \mid \_\\
&<\textbf{letra}> ::= a \mid A \mid b \mid \ldots \mid z \mid Z\\
&<\textbf{algarismo}> ::= 0 \mid 1 \mid 2 \mid \ldots \mid 8 \mid 9
\end{aligned}
$$

Notação EBNF (Exemplo avançado)

A notação EBNF extende a notação BNF com os seguinte meta-símbolos:

[ ] — parte opcional;

{ } — parte que se pode repetir 0 ou mais vezes;

( ) — precedências dentro da regra ;

“ ” — carácter a tratar como símbolo terminal *e.g.*, “$<$”.

$$<\textbf{if}> ::= \ \textit{if}\ <\textbf{condição}>\ \textit{then}\ <\textbf{instruções}>[\ \textit{else}\ <\textbf{instruções}>]\ \textit{endif}$$

$$<\textbf{identificador}> ::= \ (<\textbf{letra}> \mid \_)\{(<\textbf{letra}> \mid <\textbf{algarismo}> \mid \_)\}$$

$$<\textbf{letra}> ::= \ a \mid A \mid b \mid ... \mid z \mid Z$$

$$<\textbf{algarismo}> ::= \ 0 \mid 1 \mid 2 \mid ... \mid 8 \mid 9$$

Exercícios

Escreva uma gramática para as seguintes linguagens:

- Números binários pares
- Números binários com o máximo de 2 zeros
- Números binários começados e terminados por 1
- Números binários em que 111 é factor do número
- Números binários capicuas

Diga que tipo de gramática escreveu e se será possível convertê-la numa gramática mais restrita (tipo maior)

Gramática ambígua

Uma gramática diz-se ambígua, se é possível obter uma mesma frase a partir de duas árvores de derivação distintas.

$$\mathbf{S} \rightarrow a\mathbf{S} \mid \mathbf{S}a \mid b$$

Gramática ambígua

Uma gramática diz-se ambígua, se é possível obter uma mesma frase a partir de duas árvores de derivação distintas.

$$\mathbf{S} \rightarrow a\mathbf{S} \mid \mathbf{S}a \mid b$$

A frase ***aba*** pode ser obtida através de qualquer uma das seguintes sequências de derivação:

$$\mathbf{S} \Rightarrow a\mathbf{S} \Rightarrow a\mathbf{S}a \Rightarrow aba$$

$$\mathbf{S} \Rightarrow \mathbf{S}a \Rightarrow a\mathbf{S}a \Rightarrow aba$$

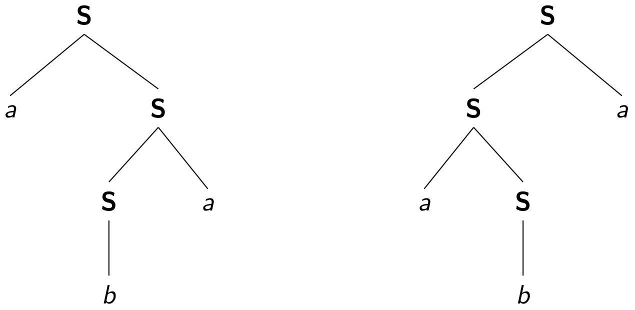
Gramática ambígua

Uma gramática diz-se ambígua, se é possível obter uma mesma frase a partir de duas árvores de derivação distintas.

$$\mathbf{S} \rightarrow a\mathbf{S} \mid \mathbf{S}a \mid b$$

A frase ***aba*** pode ser obtida através de qualquer uma das seguintes sequências de derivação:

$$\mathbf{S} \Rightarrow a\mathbf{S} \Rightarrow a\mathbf{S}a \Rightarrow aba$$
$$\mathbf{S} \Rightarrow \mathbf{S}a \Rightarrow a\mathbf{S}a \Rightarrow aba$$



Linguagem ambigua

- Quando a ambiguidade depende da gramática pode ser eliminada.
- Nos casos em que depende da linguagem, a ambiguidade não pode ser eliminada.

Linguagem ambigua

- Quando a ambiguidade depende da gramática pode ser eliminada.
- Nos casos em que depende da linguagem, a ambiguidade não pode ser eliminada.

Exemplo de linguagem inerentemente ambígua:

$$L = \left\{ a^i b^i c^j : i, j \geq 1 \right\} \cup \left\{ a^i b^j c^j : i, j \geq 1 \right\}$$

Linguagem ambigua

- Quando a ambiguidade depende da gramática pode ser eliminada.
- Nos casos em que depende da linguagem, a ambiguidade não pode ser eliminada.

Exemplo de linguagem inerentemente ambígua:

$$L = \left\{ a^i b^i c^j : i, j \geq 1 \right\} \cup \left\{ a^i b^j c^j : i, j \geq 1 \right\}$$

Esta linguagem pode ser reconhecida pela seguinte gramática:

$$\begin{aligned} \mathbf{S} &\rightarrow \mathbf{AB} \mid \mathbf{CD} \\ \mathbf{A} &\rightarrow a\mathbf{A}b \mid ab \\ \mathbf{B} &\rightarrow c\mathbf{B} \mid c \\ \mathbf{C} &\rightarrow a\mathbf{C} \mid a \\ \mathbf{D} &\rightarrow b\mathbf{D}c \mid bc \end{aligned}$$

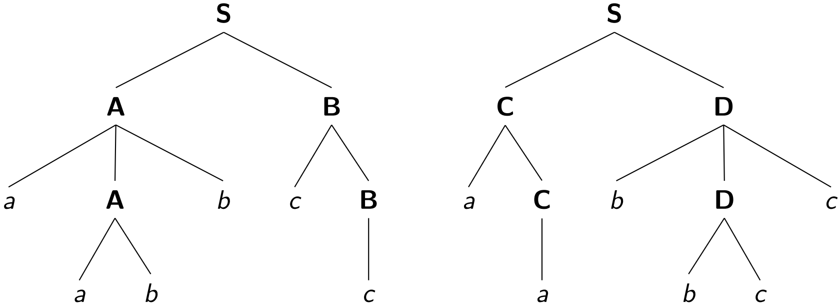
Esta linguagem é ambígua para palavras em que $i = j$

Linguagem ambigua

$$\begin{aligned}
\mathbf{S} &\rightarrow \mathbf{AB} \mid \mathbf{CD} \\
\mathbf{A} &\rightarrow a\mathbf{A}b \mid ab \\
\mathbf{B} &\rightarrow c\mathbf{B} \mid c \\
\mathbf{C} &\rightarrow a\mathbf{C} \mid a \\
\mathbf{D} &\rightarrow b\mathbf{D}c \mid bc
\end{aligned}$$

$$\mathbf{S} \Rightarrow \mathbf{AB} \Rightarrow a\mathbf{A}b\mathbf{B} \Rightarrow aabb\mathbf{B} \Rightarrow aabbc\mathbf{B} \Rightarrow aabbcc$$

$$\mathbf{S} \Rightarrow \mathbf{CD} \Rightarrow a\mathbf{CD} \Rightarrow aa\mathbf{D} \Rightarrow aab\mathbf{D}c \Rightarrow aabbcc$$



Eliminação de Ambiguidade — Dupla Recursividade

$$
\begin{aligned}
\textbf{E} &\rightarrow <\textbf{E}><\textbf{OP}><\textbf{E}> \mid <\textbf{ID}> \\
\textbf{OP} &\rightarrow + \mid \text{-} \mid \times \mid \div \\
\textbf{ID} &\rightarrow x \mid y \mid z
\end{aligned}
$$

Eliminação de Ambiguidade — Dupla Recursividade

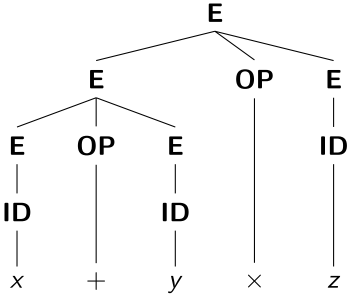
$$\begin{aligned}
\textbf{E} &\rightarrow <\textbf{E}><\textbf{OP}><\textbf{E}> \mid <\textbf{ID}> \\
\textbf{OP} &\rightarrow + \mid \text{-} \mid \times \mid \div \\
\textbf{ID} &\rightarrow x \mid y \mid z
\end{aligned}$$

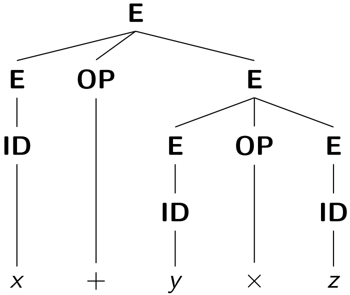
$$\begin{aligned}
<\textbf{E}> &\Rightarrow <\textbf{E}><\textbf{OP}><\textbf{E}> \Rightarrow <\textbf{E}><\textbf{OP}><\textbf{E}><\textbf{OP}><\textbf{E}> \Rightarrow \\
&\Rightarrow <\textbf{ID}><\textbf{OP}><\textbf{E}><\textbf{OP}><\textbf{E}> \Rightarrow x<\textbf{OP}><\textbf{E}><\textbf{OP}><\textbf{E}> \Rightarrow \\
&\Rightarrow x+<\textbf{E}><\textbf{OP}><\textbf{E}> \Rightarrow x+<\textbf{ID}><\textbf{OP}><\textbf{E}> \Rightarrow x+y<\textbf{OP}><\textbf{E}> \Rightarrow \\
&\Rightarrow x+y\times<\textbf{E}> \Rightarrow x+y\times<\textbf{ID}> \Rightarrow x+y\times z
\end{aligned}$$

$$\begin{aligned}
<\textbf{E}> &\Rightarrow <\textbf{E}><\textbf{OP}><\textbf{E}> \Rightarrow <\textbf{ID}><\textbf{OP}><\textbf{E}> \Rightarrow x<\textbf{OP}><\textbf{E}> \Rightarrow x+<\textbf{E}> \Rightarrow \\
&\Rightarrow x+<\textbf{E}><\textbf{OP}><\textbf{E}> \Rightarrow x+<\textbf{ID}><\textbf{OP}><\textbf{E}> \Rightarrow x+y<\textbf{OP}><\textbf{E}> \Rightarrow \\
&\Rightarrow x+y\times<\textbf{E}> \Rightarrow x+y\times<\textbf{ID}> \Rightarrow x+y\times z
\end{aligned}$$

Eliminação de Ambiguidade — Dupla Recursividade

$$\mathbf{E} \rightarrow <\mathbf{E}><\mathbf{OP}><\mathbf{E}> \mid <\mathbf{ID}>$$
$$\mathbf{OP} \rightarrow + \mid \text{-} \mid \times \mid \div$$
$$\mathbf{ID} \rightarrow x \mid y \mid z$$





Eliminação de Ambiguidade — Dupla Recursividade

Eliminação da ambiguidade retirando uma das recursividades

$$
\begin{aligned}
\textbf{E} &\rightarrow <\textbf{E}><\textbf{OP}><\textbf{F}> \mid <\textbf{F}> \\
\textbf{F} &\rightarrow <\textbf{ID}> \\
\textbf{OP} &\rightarrow + \mid \text{-} \mid \times \mid \div \\
\textbf{ID} &\rightarrow x \mid y \mid z
\end{aligned}
$$

Eliminação de Ambiguidade — Dupla Recursividade

Eliminação da ambiguidade retirando uma das recursividades

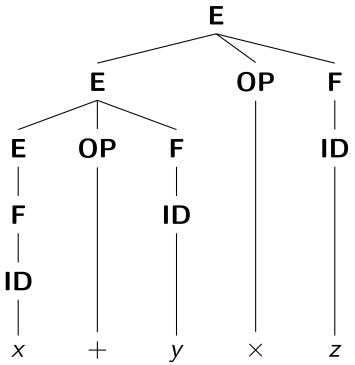
$$
\begin{aligned}
\textbf{E} &\rightarrow <\!\textbf{E}\!><\!\textbf{OP}\!><\!\textbf{F}\!> \mid <\!\textbf{F}\!> \\
\textbf{F} &\rightarrow <\!\textbf{ID}\!> \\
\textbf{OP} &\rightarrow + \mid \text{-} \mid \times \mid \div \\
\textbf{ID} &\rightarrow x \mid y \mid z
\end{aligned}
$$

$$
\begin{aligned}
<\!\textbf{E}\!> &\Rightarrow <\!\textbf{E}\!><\!\textbf{OP}\!><\!\textbf{F}\!> \Rightarrow <\!\textbf{E}\!><\!\textbf{OP}\!><\!\textbf{F}\!><\!\textbf{OP}\!><\!\textbf{F}\!> \\
&\Rightarrow <\!\textbf{F}\!><\!\textbf{OP}\!><\!\textbf{F}\!><\!\textbf{OP}\!><\!\textbf{F}\!> \Rightarrow <\!\textbf{ID}\!><\!\textbf{OP}\!><\!\textbf{F}\!><\!\textbf{OP}\!><\!\textbf{F}\!> \\
&\Rightarrow x<\!\textbf{OP}\!><\!\textbf{F}\!><\!\textbf{OP}\!><\!\textbf{F}\!> \Rightarrow x+<\!\textbf{F}\!><\!\textbf{OP}\!><\!\textbf{F}\!> \Rightarrow x+<\!\textbf{ID}\!><\!\textbf{OP}\!><\!\textbf{F}\!> \\
&\Rightarrow x+y<\!\textbf{OP}\!><\!\textbf{F}\!> \Rightarrow x+y\times<\!\textbf{F}\!> \Rightarrow x+y\times z
\end{aligned}
$$

Eliminação de Ambiguidade — Dupla Recursividade

Eliminação da ambiguidade retirando uma das recursividades

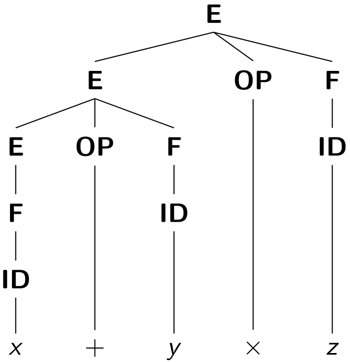
$$
\begin{aligned}
\mathbf{E} &\rightarrow <\mathbf{E}><\mathbf{OP}><\mathbf{F}> \mid <\mathbf{F}> \\
\mathbf{F} &\rightarrow <\mathbf{ID}> \\
\mathbf{OP} &\rightarrow + \mid \text{-} \mid \times \mid \div \\
\mathbf{ID} &\rightarrow x \mid y \mid z
\end{aligned}
$$



Eliminação de Ambiguidade — Dupla Recursividade

Eliminação da ambiguidade retirando uma das recursividades

$$
\begin{aligned}
\mathbf{E} &\rightarrow <\mathbf{E}><\mathbf{OP}><\mathbf{F}> \mid <\mathbf{F}> \\
\mathbf{F} &\rightarrow <\mathbf{ID}> \\
\mathbf{OP} &\rightarrow + \mid - \mid \times \mid \div \\
\mathbf{ID} &\rightarrow x \mid y \mid z
\end{aligned}
$$



A precedência dos operadores está correcta?

Precedência de Operadores

$$\mathbf{E} \rightarrow <\mathbf{E}>+<\mathbf{T}> \mid <\mathbf{E}>\text{-}<\mathbf{T}> \mid <\mathbf{T}>$$
$$\mathbf{T} \rightarrow <\mathbf{T}>\times<\mathbf{F}> \mid <\mathbf{T}>\div<\mathbf{F}> \mid <\mathbf{F}>$$
$$\mathbf{F} \rightarrow <\mathbf{ID}> \mid (<\mathbf{E}>)$$
$$\mathbf{ID} \rightarrow x \mid y \mid z$$

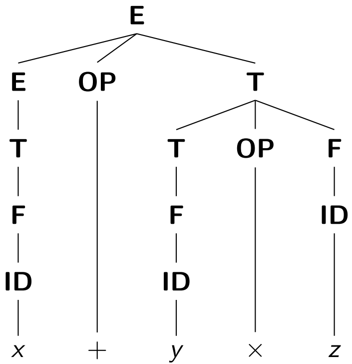
Precedência de Operadores

$$
\begin{aligned}
\textbf{E} &\rightarrow <\textbf{E}>+<\textbf{T}> \mid <\textbf{E}>\text{-}<\textbf{T}> \mid <\textbf{T}> \\
\textbf{T} &\rightarrow <\textbf{T}>\times<\textbf{F}> \mid <\textbf{T}>\div<\textbf{F}> \mid <\textbf{F}> \\
\textbf{F} &\rightarrow <\textbf{ID}> \mid (<\textbf{E}>) \\
\textbf{ID} &\rightarrow x \mid y \mid z
\end{aligned}
$$

$$
\begin{aligned}
<\textbf{E}> &\Rightarrow <\textbf{E}>+<\textbf{T}> \Rightarrow <\textbf{T}>+<\textbf{T}> \Rightarrow <\textbf{F}>+<\textbf{T}> \\
&\Rightarrow <\textbf{ID}>+<\textbf{T}> \Rightarrow x+<\textbf{T}> \Rightarrow x+<\textbf{T}>\times<\textbf{F}> \\
&\Rightarrow x+<\textbf{F}>\times<\textbf{F}> \Rightarrow x+<\textbf{ID}>\times<\textbf{F}> \\
&\Rightarrow x+y\times<\textbf{F}> \Rightarrow x+y\times<\textbf{ID}> \Rightarrow x+y\times z
\end{aligned}
$$

Precedência de Operadores

$$\mathbf{E} \rightarrow <\mathbf{E}>+<\mathbf{T}> \mid <\mathbf{E}>-<\mathbf{T}> \mid <\mathbf{T}>$$
$$\mathbf{T} \rightarrow <\mathbf{T}>\times<\mathbf{F}> \mid <\mathbf{T}>\div<\mathbf{F}> \mid <\mathbf{F}>$$
$$\mathbf{F} \rightarrow <\mathbf{ID}> \mid (<\mathbf{E}>)$$
$$\mathbf{ID} \rightarrow x \mid y \mid z$$



Ambiguidade do If

$\mathbf{S} \rightarrow if<\mathbf{C}>then<\mathbf{S}>$

$\mathbf{S} \rightarrow if<\mathbf{C}>then<\mathbf{S}>else<\mathbf{S}>$

$\mathbf{S} \rightarrow x$

$\mathbf{C} \rightarrow 0 \mid 1$

Ambiguidade do If

$$\mathbf{S} \rightarrow if{<}\mathbf{C}{>}then{<}\mathbf{S}{>}$$
$$\mathbf{S} \rightarrow if{<}\mathbf{C}{>}then{<}\mathbf{S}{>}else{<}\mathbf{S}{>}$$
$$\mathbf{S} \rightarrow x$$
$$\mathbf{C} \rightarrow 0 \mid 1$$

"*if 0 then if 1 then x else x*"

Ambiguidade do If

$$\mathbf{S} \rightarrow if < \mathbf{C} > then < \mathbf{S} >$$
$$\mathbf{S} \rightarrow if < \mathbf{C} > then < \mathbf{S} > else < \mathbf{S} >$$
$$\mathbf{S} \rightarrow x$$
$$\mathbf{C} \rightarrow 0 \mid 1$$

"*if 0 then if 1 then x else x*"

$$< \mathbf{S} > \Rightarrow if < \mathbf{C} > then < \mathbf{S} > \Rightarrow if\ 0\ then < \mathbf{S} >$$
$$\Rightarrow if\ 0\ then\ if < \mathbf{C} > then < \mathbf{S} > else < \mathbf{S} > \Rightarrow if\ 0\ then\ if\ 1\ then < \mathbf{S} > else < \mathbf{S} >$$
$$\Rightarrow if\ 0\ then\ if\ 1\ then\ x\ else < \mathbf{S} > \Rightarrow if\ 0\ then\ if\ 1\ then\ x\ else\ x$$

$$< \mathbf{S} > \Rightarrow if < \mathbf{C} > then < \mathbf{S} > else < \mathbf{S} > \Rightarrow if\ 0\ then < \mathbf{S} > else < \mathbf{S} >$$
$$\Rightarrow if\ 0\ then\ if < \mathbf{C} > then < \mathbf{S} > else < \mathbf{S} > \Rightarrow if\ 0\ then\ if\ 1\ then < \mathbf{S} > else < \mathbf{S} >$$
$$\Rightarrow if\ 0\ then\ if\ 1\ then\ x\ else < \mathbf{S} > \Rightarrow if\ 0\ then\ if\ 1\ then\ x\ else\ x$$

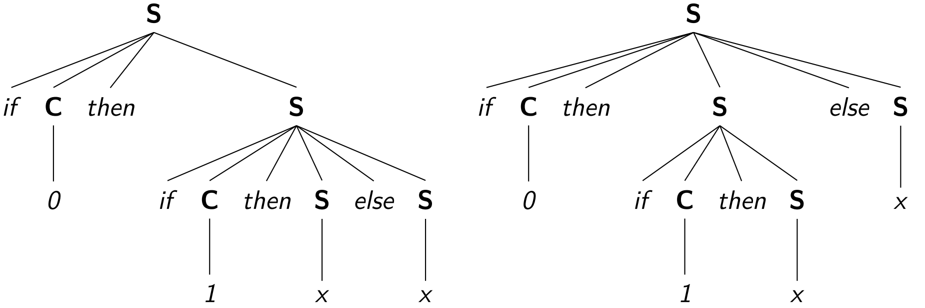
Ambiguidade do If

$\mathbf{S} \rightarrow \textit{if}<\mathbf{C}>\textit{then}<\mathbf{S}>$

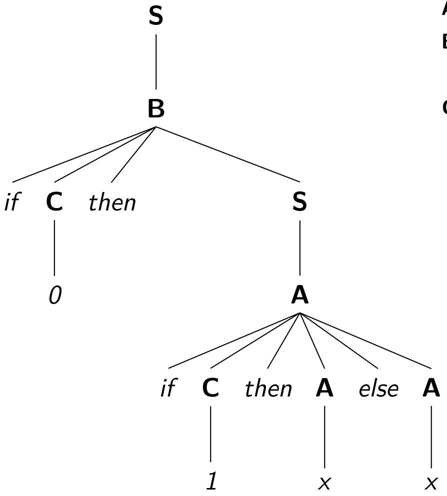
$\mathbf{S} \rightarrow \textit{if}<\mathbf{C}>\textit{then}<\mathbf{S}>\textit{else}<\mathbf{S}>$

$\mathbf{S} \rightarrow x$

$\mathbf{C} \rightarrow 0 \mid 1$



Ambiguidade do If — Resolução



$\mathbf{S} \rightarrow <\mathbf{A}> \mid <\mathbf{B}>$

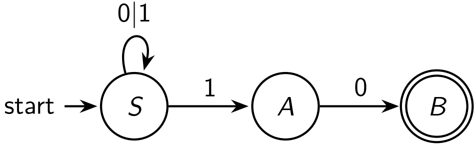
$\mathbf{A} \rightarrow \textit{if}<\mathbf{C}>\textit{then}<\mathbf{A}>\textit{else}<\mathbf{A}> \mid x$

$\mathbf{B} \rightarrow \textit{if}<\mathbf{C}>\textit{then}<\mathbf{S}> \mid$

$\quad \textit{if}<\mathbf{C}>\textit{then}<\mathbf{A}>\textit{else}<\mathbf{B}>$

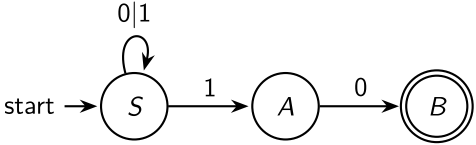
$\mathbf{C} \rightarrow \mathit{0} \mid \mathit{1}$

Conversão de Autómatos Finitos em Gramáticas



- Criar uma produção (não terminal) para cada estado;
- Para cada transição, criar uma alternativa na produção;
- Todos os estados finais podem receber a cadeia vazia.

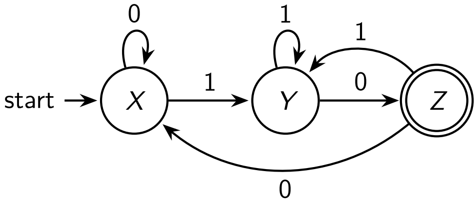
Conversão de Autómatos Finitos em Gramáticas



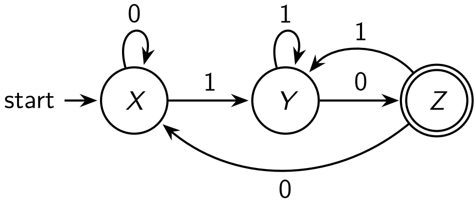
- Criar uma produção (não terminal) para cada estado;
- Para cada transição, criar uma alternativa na produção;
- Todos os estados finais podem receber a cadeia vazia.

$$\mathbf{S} \rightarrow \mathit{0}\mathbf{S} \mid \mathit{1}\mathbf{S} \mid \mathit{1}\mathbf{A}$$
$$\mathbf{A} \rightarrow \mathit{0}\mathbf{B}$$
$$\mathbf{B} \rightarrow \varepsilon$$

Conversão de Autómatos Finitos em Gramáticas



Conversão de Autómatos Finitos em Gramáticas



$$\mathbf{X} \rightarrow \mathit{0}\mathbf{X} \mid \mathit{1}\mathbf{Y}$$
$$\mathbf{Y} \rightarrow \mathit{0}\mathbf{Z} \mid \mathit{1}\mathbf{Y}$$
$$\mathbf{Z} \rightarrow \mathit{0}\mathbf{X} \mid \mathit{1}\mathbf{Y} \mid \varepsilon$$

Conversão de Gramáticas em Autómatos Finitos

- A gramática tem que ser do tipo 3 e ser linear à direita;
- Criar um estado para cada não terminal;
- Criar tansições para as produções tal que:
  - As produções do tipo $X \rightarrow aY$ dão transições do tipo $\delta(X, a) = Y$;
  - As produções do tipo $X \rightarrow \varepsilon$ implicam que X seja um estado final;
  - As produções do tipo $X \rightarrow a$ implicam a transição para um novo estado final sem transições.

Conversão de Gramáticas em Autómatos Finitos

- A gramática tem que ser do tipo 3 e ser linear à direita;
- Criar um estado para cada não terminal;
- Criar tansições para as produções tal que:
  - As produções do tipo $X \rightarrow aY$ dão transições do tipo $\delta(X, a) = Y$;
  - As produções do tipo $X \rightarrow \varepsilon$ implicam que X seja um estado final;
  - As produções do tipo $X \rightarrow a$ implicam a transição para um novo estado final sem transições.

$$\mathbf{A} \rightarrow \mathit{1}\mathbf{A} \mid \mathit{0}\mathbf{B}$$

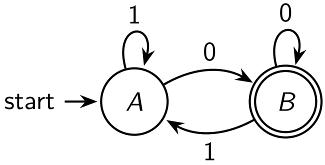
$$\mathbf{B} \rightarrow \mathit{0}\mathbf{B} \mid \mathit{1}\mathbf{A} \mid \varepsilon$$

Conversão de Gramáticas em Autómatos Finitos

- A gramática tem que ser do tipo 3 e ser linear à direita;
- Criar um estado para cada não terminal;
- Criar tansições para as produções tal que:
  - As produções do tipo $X \rightarrow aY$ dão transições do tipo $\delta(X, a) = Y$;
  - As produções do tipo $X \rightarrow \varepsilon$ implicam que X seja um estado final;
  - As produções do tipo $X \rightarrow a$ implicam a transição para um novo estado final sem transições.

$$\mathbf{A} \rightarrow 1\mathbf{A} \mid 0\mathbf{B}$$

$$\mathbf{B} \rightarrow 0\mathbf{B} \mid 1\mathbf{A} \mid \varepsilon$$



Parsers em descida recursivo

- A descida recursiva pode ser usada, para implementar *parsers* preditivos;
- Um *parser* preditivo é capaz de escolher a produção a aplicar simplesmente sabendo o **não terminal** actual e o **terminal** a ser processado;

Parsers em descida recursivo

- A descida recursiva pode ser usada, para implementar *parsers* preditivos;
- Um *parser* preditivo é capaz de escolher a produção a aplicar simplesmente sabendo o **não terminal** actual e o **terminal** a ser processado;
- Este tipo de gramáticas são chamadas de LL(1), onde:
  - o primeiro "L" indica que a frase é processada da esquerda para a direita;
  - o segundo "L" indica que se usa a derivação mais à esquerda;
  - o (1) indica o número de terminais de avanço necessários para escolher entre produções alternativas;
- Todas as gramáticas do tipo 3 podem ser convertidas em LL(1).

Parsers em descida recursivo

Para construir um *parser* em descida recursiva é necessário

- criar uma função para processar cada um dos não terminais;
- para cada não terminal, processar os *starters* e chamar as funções relativas aos não terminais que se lhes seguem;

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

Parsers em descida recursivo

```
%{
  enum{TOKEN_A,TOKEN_B,TOKEN_C,TOKEN_D,OUTRO,FIM};

  int token, nerros=0;

  void s(); /* protótipos */
  void a();
  void b();
%}

%%

a|A        return TOKEN_A;
b|B        return TOKEN_B;
c|C        return TOKEN_C;
d|D        return TOKEN_D;
.          return OUTRO;
\n         return FIM;
<<EOF>>    return FIM;

%%
```

Parsers em descida recursivo

```
void erro(char *s)
{
  nerros++;
  printf("%s<-Erro %d: %s\n",yytext,nerros,s);
}

void getToken()
{
  printf("%s",yytext);
  token=yylex();
}

int main()
{
  getToken(); /* vai buscar o 1º token */
  s();            /* chama a produção inicial */

  if (nerros==0 && token==FIM)
    printf("\nExpressão válida \n");
  else
    erro("símbolo não reconhecido (esperava a,b,c,d ou fim)");
  return 0;
}
```

Parsers em descida recursivo

```
void s() /* S-> aA | cS | dS */
{
  switch (token) {
    case TOKEN_A : getToken(); a(); break;
    case TOKEN_C :
    case TOKEN_D : getToken(); s(); break;
    default   :
         erro("símbolo não reconhecido (esperava a,c ou d)");
  }
}

void a() /* A-> aA | bB | cS | dS  */
{
  switch (token) {
    case TOKEN_A : getToken(); a(); break;
    case TOKEN_B : getToken(); b(); break;
    case TOKEN_C :
    case TOKEN_D : getToken(); s(); break;
    default   :
         erro("símbolo não reconhecido (esperava a,b,c ou d)");
  }
}
```

Parsers em descida recursivo

```
void b() /* B-> aB | bB | cB | dB | ε */
{
  switch (token) {
    case TOKEN_A :
    case TOKEN_B :
    case TOKEN_C :
    case TOKEN_D : getToken(); b(); break;
  }
  /* como pode ser vazio não dá erro */
}
```