Linguagens Formais e Autómatos / Compiladores

Análise sintática descendente

Artur Pereira / Miguel Oliveira e Silva <{artur,mos}@ua.pt>

DETI, Universidade de Aveiro

Parser recursivo-descendente: exemplo #1

$\mathcal{Q}$ Sobre o alfabeto $\{\mathtt{a}, \mathtt{b}\}$, considere linguagem

$$L_1 = \{\mathtt{a}^n\mathtt{b}^n \,:\, n \geq 0\}$$

descrita pela gramática

$$S \rightarrow \mathtt{a}\ S\ \mathtt{b} \mid \varepsilon$$

Construa um programa em que o símbolo não terminal S seja uma função recursiva que reconheça a linguagem L_1.

$\mathcal{R}$

```
int main()
{
    while (1)
    {
        printf(">> ");
        adv();
        S();
        eat('\n');
        printf("\n");
    }

    return 0;
}
```

```
void S(void)
{
    switch(lookahead)
    {
        case 'a':
            eat('a'); S(); eat('b');
            break;
        default:
            epsilon();
            break;
    }
}
```

```
void eat(int c)
{
    if (lookahead != c) error()

    if (c != '\n') adv();
}

void epsilon()
{
}

void error()
{
    printf("Unexpected symbol\n");
    exit(1);
}
```

Parser recursivo-descendente: exemplo #1

No programa anterior:

- `lookahead` é uma variável global que representa o próximo símbolo à entrada
- `adv()` é uma função que avança na entrada, colocando em `lookahead` o próximo símbolo
- `eat(`*c*`)` é uma função que verifica se no `lookahead` está o símbolo *c*, gerando erro se não estiver, e avança para o próximo
- Há duas produções da gramática com cabeça S, sendo a decisão central do programa a escolha de qual usar face ao valor do `loookahead`.
  - deve-se escolher $S \to \texttt{a}\ S\ \texttt{b}$ se o `lookahead` for `a`
  - e $S \to \varepsilon$ se o `lookahead` for `$` ou `b`

  No programa, `$`, marcador de fim de entrada, corresponde ao `\n`
- Uma palavra é aceite pelo programa se

  ```
  S(); eat($)
  ```

  não der erro.

PARSER RECURSIVO-DESCENDENTE: EXEMPLO #2

$\mathcal{Q}$ Sobre o alfabeto $\{\mathtt{a}, \mathtt{b}\}$, considere linguagem

$$L_2 \quad = \quad \{\omega \in T^* \ : \ \#(\mathtt{a}, \omega) = \#(\mathtt{b}, \omega)\}$$

descrita pela gramática

$$\begin{array}{lcl} S & \rightarrow & \varepsilon \mid \mathtt{a}\ B\ S \mid \mathtt{b}\ A\ S \\ A & \rightarrow & \mathtt{a} \mid \mathtt{b}\ A\ A \\ B & \rightarrow & \mathtt{a}\ B\ B \mid \mathtt{b} \end{array}$$

Construa um programa em que os símbolos não terminais sejas funções recursivas que reconheça a linguagem L_2.

- O programa terá 3 funções recursivas, A, B e S, semelhantes à função S do programa anterior
- Na função A, deve-se escolher $A \rightarrow \mathtt{a}$ se `lookahead` for `a` e $A \rightarrow \mathtt{b}\ A\ A$ se for `b`
- Na função B, deve-se escolher $B \rightarrow \mathtt{b}$ se `lookahead` for `b` e $B \rightarrow \mathtt{a}\ B\ B$ se for `a`
- Na função S, deve-se escolher $S \rightarrow \mathtt{a}\ B\ S$ se `lookahead` for `a`, $S \rightarrow \mathtt{b}\ A\ S$ se for `b` e $S \rightarrow \varepsilon$ se for `$`

Parser recursivo-descendente: exemplo #2a

$\mathcal{Q}$ Sobre o alfabeto $\{\mathtt{a}, \mathtt{b}\}$, considere linguagem

$$L_2 = \{\omega \in T^* : \#(\mathtt{a}, \omega) = \#(\mathtt{b}, \omega)\}$$

descrita pela gramática

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

Construa um programa em que os símbolos não terminais sejas funções recursivas que reconheça a linguagem L_{2a}.

- Escolher $S \rightarrow \varepsilon$ quando `lookahead` for `$` pode não resolver
- Por exemplo, com o `lookahead` igual a `a`, há situações em que se tem de escolher $S \rightarrow \mathtt{a}\ B$ e outras $S \rightarrow \varepsilon$
  - É o que acontece com a entrada `bbaa`

Parser recursivo-descendente: exemplo #2b

$\mathcal{Q}$ Sobre o alfabeto $\{\mathtt{a},\mathtt{b}\}$, considere linguagem

$$L_2 \quad = \quad \{\omega \in T^* \ : \ \#(\mathtt{a},\omega) = \#(\mathtt{b},\omega)\}$$

descrita pela gramática

$$\begin{array}{lcl} S & \rightarrow & \varepsilon \\ & | & \mathtt{a}\ S\ \mathtt{b}\ S \\ & | & \mathtt{b}\ S\ \mathtt{a}\ S \end{array}$$

Construa um programa em que os símbolos não terminais sejas funções recursivas que reconheça a linguagem L_{2b}

- Tal como no caso anterior, escolher $S \rightarrow \varepsilon$ quando `lookahead` for `$` pode não resolver

PARSER RECURSIVO-DESCENDENTE: EXEMPLO #3

$\mathcal{Q}$ Sobre o alfabeto $\{\texttt{a}, \texttt{b}\}$, considere linguagem

$$L_3 = \{\texttt{a}^n\texttt{b}^n \ : \ n \geq 1\}$$

descrita pela gramática

$$\begin{array}{lcl} S & \rightarrow & \texttt{a} \ S \ \texttt{b} \\ & | & \texttt{a} \ \texttt{b} \end{array}$$

Construa um programa em que o símbolo não terminal S seja uma função recursiva que reconheça a linguagem L_3.

- Como escolher entre as duas produções se ambas começam pelo mesmo símbolo?

$\mathcal{R}$

- Pôr em evidência os fatores comuns à esqueda
- Aumentar o número de símbolos de *lookahead*

Parser recursivo-descendente: exemplo #4

$\mathcal{Q}$ Sobre o alfabeto $\{\mathtt{a}, \mathtt{b}\}$, considere linguagem

$$L_4 = \{(\mathtt{ab})^n : n \geq 1\}$$

que é descrita pela gramática

$$\begin{array}{rcl} S & \rightarrow & S\ \mathtt{a}\ \mathtt{b} \\ & | & \mathtt{a}\ \mathtt{b} \end{array}$$

Construa um programa em que o símbolo não terminal S seja uma função recursiva que reconheça a linguagem L_3.

- Escolher a primeira produção cria um ciclo infinito, por causa da recursividade à esquerda
- Como lidar com a recursividade à esquerda?

Questões a resolver

$\mathcal{Q}$ Que fazer quando há prefixos comuns?

$\mathcal{R}$ Pô-los em evidência

$\mathcal{Q}$ Como lidar com a recursividade à esquerda?

$\mathcal{R}$ Transformá-la em recursividade à direita

$\mathcal{Q}$ Para que valores do *lookahead* usar uma regra $A \rightarrow \alpha$?

$\mathcal{R}$ $\texttt{\textbf{predict}}(A \rightarrow \alpha)$

Fatorização à esquerda

$$\begin{array}{rcl} S & \rightarrow & \mathtt{a}\ S\ \mathtt{b} \\ & | & \mathtt{a}\ \mathtt{b} \end{array}$$

$\Leftrightarrow$

$$S \rightarrow \mathtt{a}\ (\ S\ \mathtt{b} \,|\, \mathtt{b}\ )$$

$\Leftrightarrow$

$$\begin{array}{rcl} S & \rightarrow & \mathtt{a}\ X \\ X & \rightarrow & \mathtt{b} \\ & | & S\ \mathtt{b} \end{array}$$

Eliminação de recursividade à esquerda

- Recursividade direta

$$
\begin{array}{lcl}
A & \rightarrow & A\ \alpha \\
  & | & \beta
\end{array}
$$

$\Leftrightarrow$

$$A = \beta\alpha^n \quad n \geq 0$$

$\Leftrightarrow$

$$
\begin{array}{lcl}
A & \rightarrow & \beta\ X \\
X & \rightarrow & \varepsilon \\
  & | & \alpha\ X
\end{array}
$$

Eliminação de recursividade à esquerda

- Exemplo de recursividade direta à esquerda

$$
\begin{array}{lcl}
S & \rightarrow & S\ \mathtt{a}\ \mathtt{b} \\
  & | & \mathtt{a}\ \mathtt{b}
\end{array}
$$

$\Leftrightarrow$

$$
\begin{array}{lcl}
S & \rightarrow & \mathtt{a}\ \mathtt{b}\ X \\
X & \rightarrow & \varepsilon \\
  & | & \mathtt{a}\ \mathtt{b}\ X
\end{array}
$$

Eliminação de recursividade à esquerda

- Recursividade direta múltipla

$$\begin{array}{rcl} A & \rightarrow & A\ \alpha_1 \mid A\ \alpha_2 \mid \cdots \mid A\ \alpha_n \\ & \mid & \beta_1 \mid \beta_2 \mid \cdots \mid \beta_m \end{array}$$

$\Leftrightarrow$

$$A = (\beta_1 \mid \beta_2 \mid \cdots \mid \beta_m)(\alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n)^k \quad k \geq 0$$

$\Leftrightarrow$

$$\begin{array}{rcl} A & \rightarrow & \beta_1\ X \mid \beta_2\ X \mid \cdots \mid \beta_m\ X \\ X & \rightarrow & \varepsilon \\ & \mid & \alpha_1\ X \mid \alpha_2\ X \mid \cdots \mid \alpha_n X \end{array}$$

Eliminação de recursividade à esquerda

- Exemplo de recursividade direta múltipla à esquerda

$$\begin{array}{rcl} S & \rightarrow & S\ \mathtt{a}\ \mathtt{b} \mid S\ \mathtt{c} \\ & \mid & \mathtt{a}\ \mathtt{b} \mid \mathtt{c} \end{array}$$

$\Leftrightarrow$

$$\begin{array}{rcl} S & \rightarrow & \mathtt{a}\ \mathtt{b}\ X \mid \mathtt{c}\ S \\ X & \rightarrow & \varepsilon \\ & \mid & \mathtt{a}\ \mathtt{b}\ X \mid \mathtt{c}\ X \end{array}$$

Eliminação de recursividade à esquerda

- Aplique-se este procedimento à gramática

$$\begin{array}{lcl} S & \rightarrow & A\ \mathtt{a} \mid \mathtt{b} \\ A & \rightarrow & A\ \mathtt{c} \mid S\ \mathtt{d} \mid \varepsilon \end{array}$$

- Resultado:

$$\begin{array}{lcl} S & \rightarrow & A\ \mathtt{a} \mid \mathtt{b} \\ A & \rightarrow & S\ \mathtt{d}\ X \mid X \\ X & \rightarrow & \varepsilon \mid \mathtt{c}\ X \end{array}$$

- A recursividade não foi eliminada

$$S \Rightarrow A\,\mathtt{a} \Rightarrow S\,\mathtt{d}\,X\,\mathtt{a} \Rightarrow A\,\mathtt{a}\,\mathtt{d}\,X\,\mathtt{a}$$

Eliminação de recursividade à esquerda

$\mathcal{Q}$ Como resolver a recursividade indireta?

$$
\begin{array}{rcl}
A_1 & \rightarrow & A_2\ \alpha_1 \ \mid\ \beta_1 \\
A_2 & \rightarrow & A_3\ \alpha_2 \ \mid\ \beta_2 \\
 & \dots & \\
A_n & \rightarrow & A_1\ \alpha_n \ \mid\ \beta_n
\end{array}
$$

$\mathcal{R}$ Define-se uma ordem para os símbolos não terminais e fazem-se transformações de equivalência de modo a garantir que nenhuma produção da gramática tenha por cabeça um símbolo de ordem igual ou superior ao símbolo inicial do corpo

Eliminação de recursividade à esquerda

- Apliquemos este novo procedimento à gramática, estabelecendo a ordem S, A

$$\begin{array}{rcl} S & \rightarrow & A\ \mathtt{a} \mid \mathtt{b} \\ A & \rightarrow & A\ \mathtt{c} \mid S\ \mathtt{d} \mid \varepsilon \end{array}$$

- Substituindo, na segunda linha, S pela sua definição

$$\begin{array}{rcl} S & \rightarrow & A\ \mathtt{a} \mid \mathtt{b} \\ A & \rightarrow & A\ \mathtt{c} \mid A\ \mathtt{a}\ \mathtt{d} \mid \mathtt{b}\ \mathtt{d} \mid \varepsilon \end{array}$$

- Eliminando a recursividade à esquerda direta da segunda linha

$$\begin{array}{rcl} S & \rightarrow & A\ \mathtt{a} \mid \mathtt{b} \\ A & \rightarrow & \mathtt{b}\ \mathtt{d}\ X \mid X \\ X & \rightarrow & \mathtt{c} \mid \mathtt{a}\ \mathtt{d} \mid \mathtt{c}\ X \mid \mathtt{a}\ \mathtt{d}\ X \end{array}$$

CONJUNTO predict

$$\texttt{predict}(A \to \alpha) = \begin{cases} \texttt{first}(\alpha) & \varepsilon \notin \texttt{first}(\alpha) \\ (\texttt{first}(\alpha) - \{\varepsilon\}) \cup \texttt{follow}(A) & \varepsilon \in \texttt{first}(\alpha) \end{cases}$$

$$\texttt{first}(\alpha) = \{x \in T_\varepsilon \ : \ \alpha \ \Rightarrow^* \ x\,\omega\}$$

$$\texttt{follow}(A) = \{x \in T_\$ \ : \ S\,\$ \ \Rightarrow^* \ \gamma\,A\,x\,\omega\}$$

CONJUNTO first

ALGORITMO

$\texttt{first}(\alpha)$ {

  if $(\alpha \in T_\varepsilon)$ then

    return $\{\alpha\}$

  else if $(\alpha \in N)$ then

    return $\displaystyle\bigcup_{(\alpha\to\beta)\in P} \texttt{first}(\beta)$

  else

    $h = \text{head}(\alpha)$ # *com* $|h| = 1$

    $\beta = \text{tail}(\alpha)$

    if $\varepsilon \notin \texttt{first}(h)$ then

      return $\texttt{first}(h)$

    else

      return $(\texttt{first}(h) - \{\varepsilon\}) \cup \texttt{first}(\beta)$

}

Conjunto follow

Algoritmo

```
foreach X ∈ N
    follow(X) = ∅

$ ∈ follow(S)

do
    if (A → αB) ∈ P then
        follow(B) ⊇ follow(A)
    else      /* (A → αBβ) ∈ P */
        if ε ∈ first(β) then
            follow(B) ⊇ first(β)
        else
            follow(B) ⊇ (first(β) − {ε}) ∪ follow(A)
while algum conjunto mudou
```

- Note que $\supseteq$ significa “contém”

TABELA DE DECISÃO DE UM REC. DESCENDENTE

- Para uma gramática $G = (T, N, P, S)$ e um *lookahead* de 1, a tabela de decisão de um reconhecedor descendente corresponde à função $N \times T_\$ \rightarrow \wp(P)$, onde $\wp(P)$ representa o conjunto dos subconjuntos de P
- Esta tabela pode ser preenchida usando o seguinte algoritmo

ALGORITMO

$$
\begin{array}{l}
\textbf{foreach}\ (n, t) \in (N \times T_\$) \\
\quad T(n, t) = \emptyset \\
\textbf{foreach}\ (A \rightarrow \alpha) \in P \\
\quad \textbf{foreach}\ t \in \textbf{predict}(A \rightarrow \alpha) \\
\quad\quad \text{add}\ (A \rightarrow \alpha)\ \text{to}\ T(A, t)
\end{array}
$$

Tabela de decisão: exemplo

- Revisitemos o exemplo 1

$$
\begin{aligned}
S &\rightarrow \mathtt{a}\ S\ \mathtt{b} \\
&\mid\ \varepsilon
\end{aligned}
$$

$\mathbf{first}(\mathtt{a}\ S\ \mathtt{b}) = \{\mathtt{a}\}$

$\therefore\ \mathbf{predict}(S \rightarrow \mathtt{a}\ S\ \mathtt{b}) = \{\mathtt{a}\}$

$\mathbf{first}(\varepsilon) = \{\varepsilon\}$

$\mathbf{follow}(S) = \{\$, \mathtt{b}\}$

$\therefore\ \mathbf{predict}(S \rightarrow \varepsilon) = \{\$, \mathtt{b}\}$

Tabela de decisão

	a	b	$
S	a S b	ε	ε

- Para simplificação, optou-se por nas células apenas colocar o corpo da produção, uma vez que a cabeça é definida pela linha da tabela

Tabela de decisão: exemplo

- Revisitemos o exemplo 2

$$S \rightarrow \varepsilon \mid \mathtt{a}\ B\ S \mid \mathtt{b}\ A\ S$$
$$A \rightarrow \mathtt{a} \mid \mathtt{b}\ A\ A$$
$$B \rightarrow \mathtt{a}\ B\ B \mid \mathtt{b}$$

$\mathbf{predict}(S \rightarrow \mathtt{a}\ B\ S) = \{\mathtt{a}\}$

$\mathbf{predict}(S \rightarrow \mathtt{b}\ A\ S) = \{\mathtt{b}\}$

$\mathbf{predict}(S \rightarrow \varepsilon) = \{\$\}$

$\mathbf{predict}(A \rightarrow \mathtt{a}) = \{\mathtt{a}\}$

$\mathbf{predict}(A \rightarrow \mathtt{b}\ A\ A) = \{\mathtt{b}\}$

$\mathbf{predict}(B \rightarrow \mathtt{b}) = \{\mathtt{b}\}$

$\mathbf{predict}(B \rightarrow \mathtt{a}\ B\ B) = \{\mathtt{a}\}$

Tabela de decisão

	a	b	$
S	a B S	b A S	ε
A	a	b A A	
B	a B B	b	

- As células vazias correspondem a situações de erro

Tabela de decisão: exemplo

- Revisitemos o exemplo 2a

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

$\mathbf{predict}(S \rightarrow \mathtt{a}\ B) = \{\mathtt{a}\}$

$\mathbf{predict}(S \rightarrow \mathtt{b}\ A) = \{\mathtt{b}\}$

$\mathbf{predict}(S \rightarrow \varepsilon) = \{\mathtt{a}, \mathtt{b}, \$\}$

$\mathbf{predict}(A \rightarrow \mathtt{a}\ S) = \{\mathtt{a}\}$

$\mathbf{predict}(A \rightarrow \mathtt{b}\ A\ A) = \{\mathtt{b}\}$

$\mathbf{predict}(B \rightarrow \mathtt{b}\ S) = \{\mathtt{b}\}$

$\mathbf{predict}(B \rightarrow \mathtt{a}\ B\ B) = \{\mathtt{a}\}$

Tabela de decisão

	a	b	\$
S	$\mathtt{a}\ B\ ,\varepsilon$	$\mathtt{b}\ A\ ,\varepsilon$	ε
A	$\mathtt{a}\ \ S$	$\mathtt{b}\ A\ A$	
B	$\mathtt{a}\ B\ B$	$\mathtt{b}\ S$	

- As células $(S, \mathtt{a})$ e $(S, \mathtt{b})$ têm duas produções cada, o que torna o reconhecedor inviável para um *lookahead* de 1

Tabela de decisão: exemplo

- Revisitemos o exemplo 2b

$$\begin{array}{lcl} S & \rightarrow & \varepsilon \\ & | & \mathtt{a}\ S\ \mathtt{b}\ S \\ & | & \mathtt{b}\ S\ \mathtt{a}\ S \end{array}$$

$\mathbf{predict}(S \rightarrow \mathtt{a}\ S\ \mathtt{b}\ S) = \{\mathtt{a}\}$
$\mathbf{predict}(S \rightarrow \mathtt{b}\ S\ \mathtt{a}\ S) = \{\mathtt{b}\}$
$\mathbf{predict}(S \rightarrow \varepsilon) = \{\mathtt{a}, \mathtt{b}, \$\}$

Tabela de decisão

	a	b	$
S	$\mathtt{a}\ A\ \mathtt{b}\ S\ ,\ \varepsilon$	$\mathtt{b}\ S\ \mathtt{a}\ S\ ,\ \varepsilon$	ε

- As células $(S, \mathtt{a})$ e $(S, \mathtt{b})$ têm duas produções cada, o que torna o reconhecedor inviável para um *lookahead* de 1

Tabela de decisão: exercício

$\mathcal{E}$ Considere, sobre o alfabeto $\{\texttt{i}, \texttt{f}, \texttt{v}, \texttt{,}, \texttt{;}\}$, linguagem L_4 descrita pela gramática

$$
\begin{array}{lcl}
D & \rightarrow & T\ L\ \texttt{;} \\
T & \rightarrow & \texttt{i} \\
 & | & \texttt{f} \\
L & \rightarrow & \texttt{v} \\
 & | & \texttt{v}\ \texttt{,}\ L
\end{array}
$$

$\mathcal{Q}$ Preencha a tabela de decisão de um reconhecedor descendente, com *lookahead* de 1, que reconheça a linguagem L_4.

- Antes de calcular os conjuntos **`predict`** é necessário começar por fatorizar à esquerda, por causa das produções começadas por L

Tabela de decisão: exercício

$$
\begin{array}{lcl}
D & \rightarrow & T\ L\ ; \\
T & \rightarrow & \mathtt{i} \\
  & | & \mathtt{f} \\
L & \rightarrow & \mathtt{v}\ X \\
X & \rightarrow & \\
  & | & ,\ L
\end{array}
$$

$\mathbf{predict}(D \rightarrow T\ L\ ;) = \{\mathtt{i}, \mathtt{f}\}$
$\mathbf{predict}(T \rightarrow \mathtt{i}) = \{\mathtt{i}\}$
$\mathbf{predict}(T \rightarrow \mathtt{f}) = \{\mathtt{f}\}$
$\mathbf{predict}(L \rightarrow \mathtt{v}\ X) = \{\mathtt{v}\}$
$\mathbf{predict}(L \rightarrow \varepsilon) = \{;\}$
$\mathbf{predict}(L \rightarrow ,L) = \{,\}$

Tabela de decisão

	i	f	v	,	;	$
D	$T\ L\ ;$	$T\ L\ ;$				
T	i	f				
L			$v\ X$			
X				$,L$	ε	