Linguagens e Programação

Ana Madureira amd@isep.ipp.pt António Silva ass@isep.ipp.pt
Bruno Cunha cun@isep.ipp.pt José Marinhojsm@isep.ipp.pt
Paulo Ferreira pdf@isep.ipp.pt

Instituto Superior de Engenharia do Porto

2021/2022

Análise Léxica

- Agrupar os caracteres individuais do programa fonte em "palavras", designadas por *tokens*, que são as unidades elementares das linguagens de programação, e que serão usados pelo analisador sintáctico (*parser*)
- A sequência de caracteres que formam um ***token*** chama-se ***lexema***
- Eliminar sequências de caracteres inúteis, que não têm significado sintático (espaços, tabulações, *LF*, *CR*) e os comentários
- produzir mensagens de erro quando algum carácter ou sequência de caracteres não são reconhecidos
- localizar os *tokens* no texto fonte (nº de linha, nº de coluna)
- actuar como pré-processador (em certos compiladores)

Exemplo

...	a	[	i	n	d	]	=	1	0	*	x	1	+	y	1	...

Sequência de *tokens*:

...	ID	A_PR	ID	F_PR	IGUAL	INT	VEZES	ID	MAIS	ID	...

Sequência de lexemas:

...	a	[	ind	]	=	10	*	x1	+	y1	...

Tokens e Lexemas

Token – representa um conjunto de cadeias de entrada possível

Lexema – é uma determinada cadeia de entrada associada a um *token*

Tokens	***Lexemas***
FOR	for
IF	if
WHILE	while
NUMBER	1089, 142857, 0.2, 3.14159
IDENTIFIER	i, j, counter, StudentName
OP_PLUS	+
OP_GREATER_EQUAL	>=
OPEN_PAR	(

Reconhecimento dos Tokens

Expressões Regulares — representam padrões de cadeias de caracteres

- Os *tokens* são geralmente especificados utilizando expressões regulares
- As expressões regulares são regras bem definidas que geram um conjunto de sequências de caracteres (*strings*)
- O conjunto de *strings* gerado por uma expressão regular chama-se uma **linguagem regular**

Autómatos Finitos — forma matemática de descrever tipos particulares de algoritmos, para fazer o reconhecimento de padrões em cadeias de caracteres

Expressões Regulares – Conceitos Básicos

Alfabeto: Qualquer conjunto finito, não vazio, de símbolos.

- Um alfabeto é designado por Σ
- Considerem-se os seguintes exemplos:

$$\Sigma = \{0, 1\} \text{, alfabeto para numeros binários.}$$
$$\Sigma = \{a, b, c, \ldots, z\} \text{, o conjunto das letras minúsculas.}$$

Palavra ou cadeia: é uma sequência finita (eventualmente vazia) de símbolos do alfabeto Σ

- A palavra vazia (ε) contém zero ocorrências de símbolos
- $|W|$ designa o comprimento da palavra W.

$$|aa| = 2$$
$$|\varepsilon| = 0$$

Expressões Regulares – Conceitos Básicos

Potência de um alfabeto: Σ^k representa o conjunto de todas as palavras de comprimento k, sobre um alfabeto Σ

Por exemplo, para: $\Sigma = \{0, 1\}$:

$$\Sigma^0 = \{\varepsilon\} \text{ (contém apenas a palavra vazia)}$$
$$\Sigma^1 = \{0, 1\} \text{ (não confundir com } \Sigma\text{)}$$
$$\Sigma^2 = \{00, 01, 10, 11\}$$
$$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

Expressões Regulares — Conceitos Básicos

Fecho de um alfabeto Σ: representado por $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$, corresponde ao conjunto de todas as palavras que podem ser formadas com os símbolos de Σ, incluindo a palavra vazia ε;

Para o alfabeto $\Sigma = \{0, 1\}$:

$$\Sigma^* = \{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$$

Expressões Regulares — Conceitos Básicos

Concatenação de palavras: Justapondo duas palavras α e β será obtido $\alpha\beta$, uma palavra formada pelos símbolos de α seguidos dos símbolos de β.

A operação de concatenação de palavras possui as seguintes propriedades:

- A **palavra vazia** ε constitui o **elemento identidade** (neutro) da operação, isto é, para qualquer palavra α:

$$\alpha\varepsilon = \varepsilon\alpha = \alpha$$

- **é associativa**, ou seja, para quaisquer palavras α, β e γ:

$$(\alpha\beta)\gamma = \alpha(\beta\gamma)$$

- **não é comutativa**, ou seja, $\alpha\beta \neq \beta\alpha$, sempre que $\alpha \neq \beta$.

$$\alpha = aa$$
$$\beta = b$$
$$\alpha\beta = aab \neq baa = \beta\alpha$$

Expressões Regulares — Conceitos Básicos

- O comprimento é **aditivo** relativamente à concatenação, $|\alpha\beta| = |\alpha| + |\beta|$
- Duas palavras são **iguais** sempre que uma for a cópia, símbolo a símbolo, da outra;
- Dadas duas palavras α e β:
  - a palavra α é **prefixo** da palavra $\alpha\beta$,
  - a palavra β é **sufixo** da palavra $\alpha\beta$;
- Diz-se que a palavra α é parte (**subpalavra** ou **factor**) da palavra β sempre que existam as palavras γ e δ , tais que $\beta = \gamma\alpha\delta$;

Expressões Regulares — Conceitos Básicos

- Para qualquer número natural n e qualquer palavra α, designa-se por α^n a **potência** de ordem n
  Por exemplo, para o alfabeto $\Sigma = \{a, b, c, \ldots, z\}$ e $\alpha = ab$ teremos:

  $$\alpha^1 = ab$$
  $$\alpha^2 = abab$$
  $$\alpha^3 = ababab$$

  Note-se que $(ab)^3 = ababab$, enquanto que $ab^3 = abbb$. Por convenção para qualquer palavra α temos que $\alpha^0 = \varepsilon$;

- Designa-se por α^R a **palavra inversa** de α, ou seja, se $\alpha = ab$ então $\alpha^R = ba$.

  $$(\alpha^R)^R = \alpha$$
  $$(\alpha^R)^n = (\alpha^n)^R \text{para qualquer } n$$

Linguagens Formais

- Uma linguagem formal é um conjunto de cadeias de símbolos (palavras) de um determinado alfabeto
- Uma linguagem é finita se as suas frases formam um conjunto finito (caso contrário, a linguagem é infinita)
- Uma linguagem infinita precisa ser definida através de uma representação finita

  Por exemplo, a linguagem dos números naturais menores que 10 é finita, e pode ser representada, literalmente, como:

  $$L = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

  Já a linguagem dos números naturais como um todo não é uma linguagem finita, já que existem infinitos números naturais

Linguagens Formais

Uma linguagem L sobre um alfabeto Σ é um conjunto de palavras de Σ^*, isto é, é um subset de Σ^* ($L \subseteq \Sigma^*$)

Uma linguagem L diz-se que:

- L pode não incluir todas as palavras de Σ^*, nem sequer as suas palavras incluírem todos os símbolos de Σ;
- Uma **linguagem vazia**, que não contém palavras, designa-se por $\emptyset$.

  L=$\emptyset$ é diferente de $L = \{\varepsilon\}$
- A linguagem designa-se como **completa** se coincide com a totalidade do conjunto Σ^*;
- Se L e M são linguagens de alfabeto Σ, a **concatenação** de L com M é uma linguagem de alfabeto Σ, definida por

$$LM = \{\alpha\beta \,|\, \alpha \in L \text{ e } \beta \in M\}$$
$$\{a, ab\}\,\{b, ba\} = \{ab, aba, abb, abba\}$$

Linguagens Formais

- As **potências** L^n de uma linguagem L são definidas indutivamente por:

$$L^0 = \{\varepsilon\}$$
$$L^{n+1} = LL^n, \text{para } n \geq 1$$
$$L^n = \{x_1 x_2 \dots x_n | x_i \in L, 1 \leq i \leq n\}$$

Isto é, a linguagem das palavras obtidas por concatenação de n palavras de L.

para $L = \{ab, aab\}$:

$$\{ab, aab\}^0 = \{\varepsilon\}$$
$$\{ab, aab\}^1 = \{ab, aab\}$$
$$\{ab, aab\}^2 = \{abab, abaab, aabab, aabaab\}$$
$$\{ab, aab\}^3 = \{ababab, ababaab, abaabab, aababab, abaabaab, aababaab, \dots\}$$

Linguagens Formais

- O **fecho de *Kleene*** de uma linguagem L é a reunião de todas as potências finitas de L.

$$L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \ldots = \cup_{n \geq 0} L^n$$

é conjunto das palavras que se podem formar por concatenação de zero ou mais palavras de L

Σ^*, não é mais do que o fecho de *Kleene* de Σ.

$$\begin{aligned}
\{1\}^* &= \{\varepsilon, 1, 11, 111, 1111, 11111, \ldots\} = \{1^n | n \in \mathbb{N}\} \\
\{01\}^* &= \{\varepsilon, 01, 0101, 010101, 01010101, 0101010101, \ldots\} \\
\{000\}^* &= \{\varepsilon, 000, 000000, 000000000, \ldots\} = \left\{0^{3n} | n \in \mathbb{N}\right\} \\
\{000, 00000\}^* &= \{\varepsilon, 000, 00000, 000000, 00000000, 000000000, \ldots\} \\
&= \{\varepsilon, 000, 00000, 000000\} \cup \{0^n | n \geq 8\} \\
\{0, 1\}^* &= \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, \ldots\}
\end{aligned}$$

Linguagens Formais

- O **fecho positivo** da linguagem L, definido por $L^+ = L^1 \cup L^2 \cup L^3 \cup \ldots$.
  - Se $\varepsilon \in L$ então $L^+ = L^*$
  - Se $\varepsilon \notin L$ então $L^+ = L^* \setminus \{\varepsilon\}$

- Toda a expressão regular representa uma linguagem regular;

- Toda a linguagem regular é representável por alguma expressão regular.

  Para o alfabeto binário $\Sigma = \{0, 1\}$, os seguintes sets são linguagens:
  - $\emptyset$ — linguagem vazia;
  - $\{\varepsilon\}$ — linguagem constituída pela palavra nula;
  - $\{0\}$ — linguagem constituída por uma única palavra "0";
  - $\{00, 01, 10, 11\}$ — linguagem constituída por palavras de comprimento 2.

Equivalência entre Expressões Regulares e Linguagens

O conjunto de Expressões Regulares sobre um alfabeto Σ e o conjunto das linguagens por elas descritas são definidos indutivamente por:

- ε é uma expressão regular sobre o alfabeto Σ e descreve a linguagem $\{\varepsilon\}$, $L(\varepsilon) = \{\varepsilon\}$;
- $\emptyset$ é uma expressão regular sobre o alfabeto Σ e descreve a linguagem vazia $\emptyset$ isto é, $L(\emptyset) = \emptyset$;
- Se $a \in \Sigma$ então a é uma expressão regular sobre Σ e descreve a linguagem $\{a\}$, isto é, $L(a) = \{a\}$;

Operações sobre ERs/LF

- Se r e s são Expressões Regulares sobre Σ que descrevem as linguagens $L(r)$ e $L(s)$:

$$\begin{aligned} \text{União} &\longrightarrow r|s = L(r) \cup L(s) \\ \text{Concatenação} &\longrightarrow rs = L(r)L(s) \\ \text{Fecho de Kleene} &\longrightarrow r^* = L(r)^* \end{aligned}$$

Se r e s que permitem reconhecer as palavras, $\{aa, ab\}$ e $\{bb, ba\}$:

União

$r|s = L(r) \cup L(s) = \{aa, ab, bb, ba\}$

Concatenação

$rs = L(r)L(s) = \{aabb, aaba, abbb, abba\}$

Fecho de Kleene

$r^* = L(r)^* = \{\varepsilon, aa, ab, aaaa, aaab, abaa, abab, aaaaaa, aaaaab, \ldots\}$

Expressões Regulares/Linguagens Formais

- Qualquer expressão regular r representa uma linguagem que se designa por $L(r)$;
- Diz-se que duas Expressões Regulares são **equivalentes** se, e só se, as linguagens por elas descritas são iguais.

Expressões Regulares — Precedências

A precedência para as operações união ($|$), concatenação ($\cdot$) e fecho de Kleene (*) correspondem às usadas nas expressões aritméticas para a **adição**, a **multiplicação** e a **potenciação**, respectivamente.

fecho de Kleene $>$ concatenação $>$ união.

Considerando as "ERs" r, s e t. Então a "ER" $rs^*|t$ será avaliada da forma seguinte:

1º Passo — avaliação da potência (fecho de Kleene)

$$rs^*|t \Rightarrow r\{\varepsilon, s, ss, sss, \ldots\}|t$$

2º Passo — avaliação da concatenação

$$rs^*|t \Rightarrow \{r\varepsilon, rs, rss, rsss, \ldots\}|t$$

sendo que $r\varepsilon = r$

3º Passo — avaliação da união

$$rs^*|t \Rightarrow \{r, rs, rss, rsss, \ldots, t\}$$

Exemplos de Expressões Regulares

Seja $\Sigma = \{0, 1\}$

expressão regular	**Linguagem Descrita**
$0\vert 1$	$\{0, 1\}$
0^*	$\{\varepsilon, 0, 00, 000, 0000, 00000, \ldots\}$
0^*11	$\{11, 011, 0011, 00011, 000011, 0000011, \ldots\}$
$(01)^*$	$\{\varepsilon, 01, 0101, 010101, 01010101, 0101010101, \ldots\}$
$(0\vert 1)^*$	$\{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, \ldots\}$
$1^*0(0\vert 1)^*$	$\{0, 10, 01, 00, 000, 100, 010, 001, 110, 101, 011, \ldots\}$
$(0\vert 1)^*0$	sequências de palavras terminadas (ou com sufixo) em 0
$(0\vert 1)^*1$	sequências de palavras terminadas (ou com sufixo) em 1
$(0\vert 1)^*101(0\vert 1)^*$	sequência que contém o factor ou subpalavra 101

Propriedades das Expressões Regulares

Sejam u, v e x Expressões Regulares. Tem-se que:

$$u|v = v|u$$

$$u|u = u$$

$$(u|v)|x = u|(v|x)$$

$$u\varepsilon = \varepsilon u = u$$

$$(uv)x = u(vx)$$

$$u(v|x) = uv|ux$$

$$(u|v)x = ux|vx$$

$$\varepsilon^* = \varepsilon$$

Propriedades das Expressões Regulares

Sejam u, v e x Expressões Regulares. Tem-se que:

$$u^* = u^*u^* = (u^*)^* = u^*|u^*$$

$$u^* = \varepsilon|u^* = (\varepsilon|u)^* = (\varepsilon|u)u^* = \varepsilon|uu^*$$

$$u^* = (u|\ldots|u^k)^*, k \geq 1$$

$$u^* = \varepsilon|u|\ldots|u^{k-1}|u^k u^*, k > 1$$

$$u^*u = uu^*$$

$$(u|v)^* = (u^*|v^*)^* = (u^*v^*)^* = (u^*v)^*u^* = u^*(vu^*)^*$$

$$u(vu)^* = (uv)^*u$$

$$(u^*v)^* = \varepsilon|(u|v)^*v$$

$$(uv^*)^* = \varepsilon|u(u|v)^*$$

Expressões Regulares — Aplicação prática *FLEX*

`x`	O carácter "x"
`.`	Qualquer carácter excepto mudança de linha
`\n`	mudança de linha
`[xyz]`	Um dos caracteres "x", "y" ou "z"
`xyz`	A cadeia de caracteres "xyz"
`[a-zA-Z]`	Um dos caracteres no intervalo de "a" a "z" ou de "A" a "Z"
`[-+*/]`	Qualquer um dos operadores "-", "+", "*" ou "/", sendo que o símbolo "-" tem de aparecer em primeiro lugar dada a possibilidade de ambiguidade com a definição de intervalo
`[abj-oZ]`	Um dos caracteres "a", "b" ou de "j" a "o" ou "Z"
`[^A-Z\n]`	Qualquer carácter excepto no intervalo de "A" a "Z" ou mudança de linha
`r*`	O carácter "r" zero ou mais vezes
`r+`	O carácter "r" uma ou mais vezes
`r?`	O carácter "r" zero ou uma vez

Expressões Regulares — Aplicação prática *FLEX*

`r{2,5}` O carácter "r" repetido de duas a cinco vezes
`r{2,}` O carácter "r" repetido pelo menos duas vezes
`r{4}` O carácter "r" repetido exactamente quatro vezes
`{macro}` Substituição/Expansão da macro definida anteriormente
`(r)` O carácter "r", sendo que os parêntesis permitem estipular precedências
`xyz*` A sequência "xy" seguida de zero ou mais "z"s
`(xyz)*` A sequência "xyz" repetida zero ou mais vezes
`r|s` O carácter "r" ou "s" (alternativa)
`^r` O carácter "r" apenas se no início da linha
`r$` O carácter "r" apenas se no final da linha (não consome o \n)
`^xyz$` Uma linha que contém apenas a cadeia de caracteres "xyz"
<<EOF>> Fim de ficheiro

Expressões Regulares — Exemplos

- Definição de um identificador
  - letra → [A-Za-z]
  - dígito →[0-9]
  - identificador → {letra} ({letra}|{digito})*

- Definição de um número com parte decimal
  - dígito → [0-9]
  - dígitos → {dígito}+
  - parteFraccionária → (\. {dígitos})?
  - expoente → (E [+-]? {dígitos})?
  - num → {dígitos} {parteFraccionária} {expoente}

Expressões Regulares — Exercícios

- Escreva uma expressão regular para cada um dos seguintes casos, usando $\Sigma = \{0, 1\}$:
  - Sequência com sufixo 1
  - Sequência com prefixo 1
  - Sequência que contém dois 1s
  - Sequência que contém pelos menos três 1s e um 0
  - Sequência que contém o factor 01
  - Sequência que começa e termina com o mesmo algarismo