

Compiladores

(cap 3 – Análise Léxica: Introdução,
Revisão LFA)

Análise Léxica

- A primeira fase da compilação
- Recebe os caracteres de entrada do programa e os converte em um fluxo de tokens
- Tokens são unidades lógicas que representam um ou mais caracteres
 - Cada palavra-chave é um token. Ex: **then, begin, integer**
 - Cada identificador é um token. Ex: **a, soma, var1**
 - Cada constante é um token. Ex: **123, 123.456, 1.2E3**
 - Cada sinal é um token. Ex: **(, <, <=, +**

Análise Léxica

- A análise léxica é, usualmente, invocada pelo parser cada vez que um novo token é necessário
- A análise léxica é uma fase que processa caracter por caracter. Assim, velocidade é um ponto crítico no projeto de analisadores léxicos
- A análise léxica é dividida em dois processos (executados nesta ordem):
 - Scanning: remove comentários, remove espaços desnecessários
 - Análise léxica: agrupa os caracteres em tokens

Análise Léxica

Token, Padrão e Lexema

- **Token:** é um par consistindo de um nome de token e um valor de atributo opcional. O nome do token é um símbolo abstrato representando um tipo de unidade léxica. Ex: palavra-chave, identificador, número, etc.
- **Padrão:** é uma descrição da forma que os lexemas de um token podem tomar. No caso de uma palavra-chave como um token, o padrão é a seqüência de caracteres que formam a palavra-chave.
- **Lexema:** seqüência de caracteres no programa fonte que associa um padrão a um token e é uma instância do token

Análise Léxica

- Exemplo: `printf("Total = %d\n", score);`
- `printf` e `score` são lexemas associando o padrão para o token `id`, `"Total = %d\n"` é um lexema que associa o padrão `literal`
- Outros exemplos:

Token	Lexemas
if	if
else	else
comparison	<=, !=
id	pi, score, D2
number	3.14159, 0, 6.02e23
literal	"core dumped"

Análise Léxica

- Baseando-se na tabela de tokens e lexemas, determine os lexemas do código a seguir:

```
float quadradolimitado(x) float x{  
/* retorna o quadrado limitado de x */  
    float retorno;  
    if ((x<=-10.0) || (x>=10.0))  
        retorno=100;  
    else  
        retorno=x*x;  
    return retorno;  
}
```

Token	Lexemas
if	if
else	else
comparison	<=, !=
id	pi, score, D2
number	3.14159, 0, 6.02e23
literal	“core dumped”

Análise Léxica

Questões de Implementação

- Implementação típica: uma função que devolve o código do próximo símbolo de entrada e, eventualmente, o seu valor
- Algumas linguagens apresentam dificuldades particulares, como o uso de palavras-chave (mas não reservadas) como identificadores
- Alternativas para implementação:
 - Abordagens ad hoc
 - Ferramentas automáticas: lex, flex, Jlex, Jflex, etc.

Análise Léxica

Abordagem Ad Hoc

- Se o próximo caractere é um símbolo especial; verifique se pode ter mais de um caractere (ex: <=) e devolva o seu código
- Se o próximo símbolo é um dígito (ou, símbolo de sinal), tratar o resto do número e retornar o código (inteiro, real, double, etc.) e o valor
- Se o próximo símbolo pode iniciar um identificador, trate o resto dele; se for uma palavra reservada, devolva o código correspondente; senão, devolva o código de identificador e o valor da cadeia
- Se o próximo símbolo indica o início de uma cadeia, trate o resto dela e devolva o código de cadeia e o seu valor
- Para fins de mensagens de erro e de depuração, mantenha variáveis convenientes com o nome do arquivo, o número da linha, posição na linha, etc.

Análise Léxica

Erros léxicos

- É difícil para o analisador léxico informar que há erros no código fonte sem o auxílio de outros componentes
- Exemplo: `f i (a == f (x))`
 - O léxico não consegue determinar se `f i` é um erro de digitação da palavra-chave `if` ou se é um identificador (um nome de função)
- Mas, e se o léxico não consegue determinar qual padrão mapeia para a entrada atual?
- Nesse caso, uma estratégia (*panic-mode*) é apagar os caracteres da seqüência até que o léxico possa determinar um token válido
- Outras ações possíveis são: inserir um caracter válido, substituir um caracter por outro, transpor dois caracteres adjacentes.

Gramáticas e Linguagens - Revisão

- Alfabeto ou vocabulário: conjunto finito e não vazio de símbolos;
 - $\Sigma^1 = \{a, b\}$ (conjunto de duas letras)
 - $\Sigma^2 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ (conjunto de dígitos decimais)
 - $\Sigma^3 = \{\text{if, then, else, while, id, :=, <, >, =, ...}\}$ (conjunto de símbolos de uma linguagem de programação: cada símbolo é considerado um átomo)
- Se $\alpha = \sigma^1 \sigma^2 \dots \sigma^n$ com $\sigma^i \in \Sigma$, então α é uma cadeia sobre o alfabeto Σ , de comprimento n ; Exemplo: 'abaa' é uma cadeia sobre Σ^1 , de comprimento 4
- O símbolo ε (no texto λ) denota a cadeia vazia de comprimento zero

Gramáticas e Linguagens - Revisão

- Dada uma cadeia α :
 - $\alpha^0 = \epsilon$
 - $\alpha^n = \alpha^{n-1} \alpha$ para $n > 0$ (concatenação n vezes)
- Dado um alfabeto Σ
 - Σ^* é o conjunto de todas as cadeias finitas sobre Σ
 - Σ^+ é o conjunto de todas as cadeias finitas sobre Σ , exceto a cadeia vazia
- Uma linguagem sobre Σ é um subconjunto de Σ^*
- Dadas duas linguagens L^1 e L^2 sobre um alfabeto Σ , a concatenação (ou o produto) é dado por $L^1 L^2 = \{\alpha\beta \mid \alpha \in L^1 \text{ e } \beta \in L^2\}$

Gramáticas e Linguagens - Revisão

- Dada uma linguagem L
 - $L^0 = \{\epsilon\}$
 - $L^n = L^{n-1}L$ para $n > 0$
 - $L^* = \bigcup_{n \geq 0} L^n$
 - $L^+ = \bigcup_{n \geq 1} L^n$
- Dada uma cadeia a , a notação será estendida para:
 - $a^* = \{a\}^*$
 - $a^+ = \{a\}^+$

Gramáticas e Linguagens - Revisão

- Exemplo: gramática de expressões simples
 - $E \leftarrow a$
 - $E \leftarrow b$
 - $E \leftarrow E + E$
 - $E \leftarrow E * E$
 - $E \leftarrow (E)$
 - Ou sob a forma abreviada: $E \leftarrow a \mid b \mid E + E \mid E * E \mid (E)$
- Vocabulário terminal T : $\{a, b, +, *, (,)\}$
- Vocabulário não-terminal N : $\{E\}$
- Símbolo inicial S (raíz): E
- Produção P (exemplo): $E \leftarrow E * E$
- Gramática livre de contexto: $G = (T, N, P, S)$

Gramáticas e Linguagens - Revisão

- Caso particular de linguagens livres de contexto
- Podem ser descritas por gramáticas cujas produções têm uma forma particular; exemplo:

- $S \leftarrow a$		$S \leftarrow a$
- $S \leftarrow b$	ou	$S \leftarrow b$
- $S \leftarrow aS$		$S \leftarrow Sa$
- $S \leftarrow bS$		$S \leftarrow Sb$

- Todas as ocorrências de símbolos não terminais estão no fim (ou no início) dos lados direitos das produções
- As duas gramáticas descrevem a linguagem que contém todas as cadeias finitas e não vazias sobre o alfabeto $\{a, b\}$; exemplo, usando a primeira gramática:

$S \rightarrow aS \rightarrow abS \rightarrow abbS \rightarrow abbaS \rightarrow abbab$

Gramáticas e Linguagens - Revisão

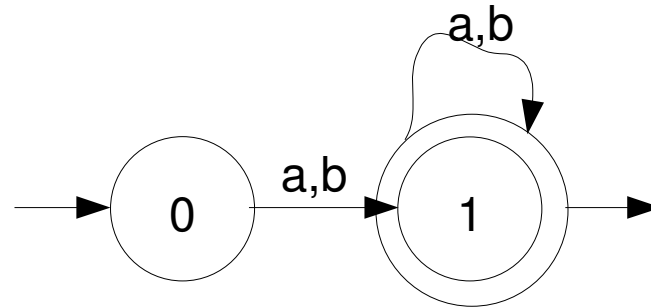
- Expressões regulares possibilitam descrever, de forma simples, linguagens regulares
- Dado um alfabeto Σ :
 - para todo $\sigma \in \Sigma$, “ σ ” é uma expressão regular que denota a linguagem $\{\sigma\}$
 - se “ α ” e “ β ” são duas expressões regulares denotando as linguagens L_α e L_β , então “ $\alpha\beta$ ” é uma expressão regular que denota a linguagem produto $L_\alpha L_\beta$
 - Se “ α ” e “ β ” são duas expressões regulares denotando as linguagens L_α e L_β , então “ $\alpha|\beta$ ” é uma expressão regular que denota a linguagem união $L_\alpha \cup L_\beta$
 - Se “ α ” é uma expressão regular que denota a linguagem L_α então “ α^* ” é uma expressão regular que denota a linguagem L_α^*
 - Se “ α ” é uma expressão regular que denota a linguagem L_α então “ (α) ” é uma expressão regular que denota a mesma linguagem

Gramáticas e Linguagens - Revisão

- Qual a linguagem descrita pelos exemplos a seguir?
 - Exemplo 1: $(a|b)(a|b)^*$
 - Exemplo 2: $a(aa|ab|ba|bb)^*b$
 - Exemplo 3: $(a|b|c|d|\dots|z|)(a|b|c|d|\dots|z|0|1|\dots|9|)^*$

Gramáticas e Linguagens - Revisão

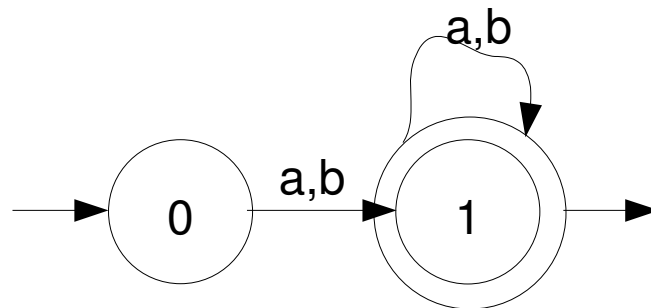
- É possível demonstrar que expressões regulares descrevem a mesma classe de linguagens aceitas por autômatos finitos
- Exemplo1: o seguinte autômato aceita a linguagem descrita por $(a|b)(a|b)^*$



- Por convenção, o estado inicial tem o rótulo 0; o círculo duplo indica estado final (aceitação)

Gramáticas e Linguagens - Revisão

- Exemplo1: o seguinte autômato aceita a linguagem descrita por $(a|b)(a|b)^*$



L0: if inchar in {a,b} goto L1
 reject()

L1: advance()
 if inchar in {a,b} goto L1
 accept()

- Como seria um autômato para reconhecer a seguinte expressão regular: $a(aa|ab|ba|bb)^*b$?