

– INF01147 –
Compiladores

Análise Léxica Expressões Regulares

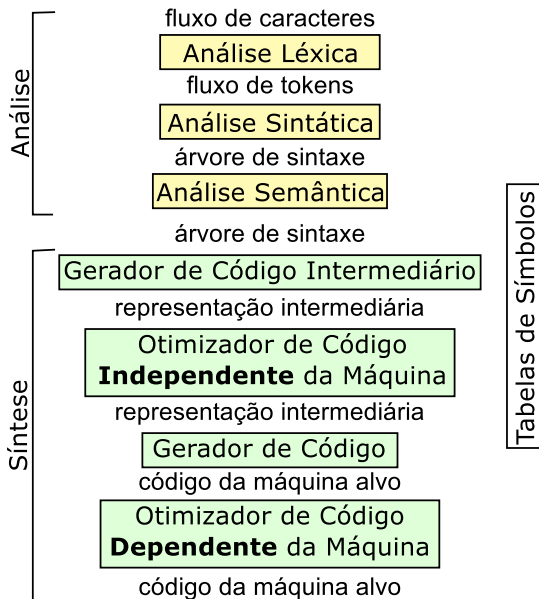
Prof. Lucas M. Schnorr
– Universidade Federal do Rio Grande do Sul –



Revisão

- ▶ O que são linguagens de programação de alto nível?
- ▶ O que é um compilador?
- ▶ Qual a diferença entre compilador e interpretador?
- ▶ Porque programas compilados são mais rápidos?
- ▶ Quais as duas etapas principais na compilação?
- ▶ Essas duas etapas estão presentes em um interpretador?

Revisão – Estrutura de um **Compilador**

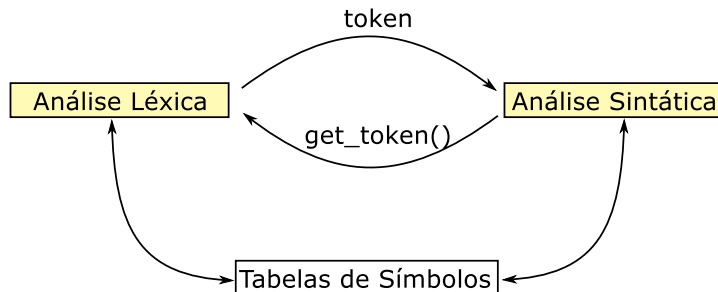


Plano da Aula de Hoje

- ▶ **Motivação** do uso de Expressões Regulares (ER)
 - ▶ Com definições: linguagens, tokens, lexemas
- ▶ **Regras de Formação** e Exemplos de ER
 - ▶ Operações
 - ▶ Exemplo de uso no Linux: `grep`
- ▶ **Autômatos Finitos** e ER
 - ▶ Definições e Exemplos
 - ▶ Construção de um autômato reconhecedor de ER

Análise Lexical

- ▶ Identificar sequências significativas na entrada: **lexemas**
- ▶ Quando um lexema é identificado, gera um **token**
- ▶ Funções adicionais
 - ▶ Começa a construção da tabela de símbolos
 - ▶ Detecção de erros léxicos, gerando mensagens de erro



Por que uma Análise Léxica?

- ▶ **Simplicidade de Projeto**
 - ▶ Simplifica o analisar sintático
 - ▶ Projeto de Linguagem mais limpo
- ▶ **Eficiência**
 - ▶ Aplicação de técnicas especializadas
 - ▶ *Buffering* → melhora o desempenho
- ▶ **Portabilidade**
 - ▶ Peculiaridades da entrada são isoladas (formato, codificação)

Análise Léxica – Vocabulário Básico

▶ Token

- ▶ Um token é composto de: **<nome-token, valor-atributo>**
- ▶ **nome-token** → representa **tipo** de unidade léxica
- ▶ **valor-atributo** → valor ou referência a tabela de símbolos
- ▶ Exemplos
 - ▶ Palavras-chave
 - ▶ Operadores
 - ▶ Identificadores
 - ▶ Constantes
 - ▶ Literais
 - ▶ Símbolos

▶ Padrão

- ▶ Descrição da forma da instância de um token

▶ Lexema

- ▶ Caracteres na entrada que casam com um padrão de token
- ▶ Instância de um token

Exemplos de Token, Padrão e Lexema

Token	Padrão	Exemplos de Lexema
if	caracteres i,f	if
else	caracteres e,l,s,e	else
comparação	<,<=,=,>,>=	< ou <= ou =
id	letras e dígitos	pi, score, D2
número	constante numérica	3.14159
literal	string cercada por “s	“core dumped”

Visão rápida da Implementação

- ▶ Utilização de buffers de entrada
 - ▶ Normalmente um **par de buffer**
 - ▶ Cada um do tamanho de um bloco de disco
- ▶ Dois apontadores
 - ▶ *lexemeBegin*
 - ▶ *forward*
- ▶ Uso de **sentinelas** – **eof**
 - ▶ Fim do buffer
 - ▶ Fim da entrada
- ▶ Exemplo: $pos = i + r * 60$
- ▶ **Como reconhecer padrões?**

Como definir padrões?

- ▶ Enumerar os valores possíveis: $<$, \leq , $=$, \geq , $>$
→ Complexo e não eficaz
- ▶ Usar conceitos de linguagens formais
- ▶ Quatro categorias de linguagens (segundo Chomsky)
 - ▶ **Regulares** (3) → Autômatos de Estados Finitos
 - ▶ Livres de Contexto (2) → Autômato com pilha
 - ▶ Sensíveis ao Contexto (1) → Máquina de Turing + Fita Infin.
 - ▶ Recursivamente Enumeráveis (0) → Máquina de Turing
- ▶ Em linguagens de programação
 - ▶ Representadas por Gramáticas
 - ▶ Reconhecidas por Autômatos

Definições de Linguagens Formais

- ▶ **Símbolo**
 - ▶ Entidade Abstrata
Letras, Dígitos, Caracteres Especiais
- ▶ **Alfabeto (Σ)** – sigma
 - ▶ Um conjunto finito de símbolos
 $\Sigma = \{0, 1\}$
- ▶ **String (cadeira, frase)**
 - ▶ Sequência finita de símbolos de um determinado alfabeto
- ▶ **String Vazia (ϵ)** – epsilon
 - ▶ String com nenhum símbolo
- ▶ **Linguagem Formal (Σ^*)**
 - ▶ Conjunto de todas as possíveis strings de um alfabeto

Expressão Regular (ER)

- ▶ Como definir uma ER?
 - ▶ ER básicas \rightarrow definida por literais
 - ▶ ER complexas \rightarrow combinação de ER básicas
- ▶ Algumas regras para formação de palavras válidas
 - ▶ Concatenação: xy (x seguido de y)
 - ▶ Alternância: $x|y$ (x ou y)
 - ▶ Repetição: x^* (x repetido 0 ou mais vezes)
 - ▶ Repetição: x^+ (x repetido 1 ou mais vezes)

Precedência em Expressões Regulares

- ▶ Diminuindo a precedência, a partir de
 - ▶ Operador unário $*$, $+$
 - ▶ Concatenação
 - ▶ Operador $|$
- ▶ Sempre com associatividade a esquerda
- ▶ **Parênteses** para resolver casos ambíguos
- ▶ Exemplos

$(a)|((b)*(c))$ equivalente a:
 $a|b*c$

$a(b|c)d$ diferente de:
 $ab|cd$

Exemplos de Expressões Regulares

- ▶ **ab** representa o conjunto $\{ ab \}$
- ▶ $(a|b) \rightarrow \{ a, b \}$
- ▶ $(a|b)(a|b) \rightarrow \{ aa, ab, ba, bb \}$
- ▶ $a^* \rightarrow \{ \epsilon, a, aa, aaa, aaaa, \dots \}$
- ▶ $(a|b)^* \rightarrow$ todas as strings com a e b
- ▶ $(a|b)^*aa(a|b) \rightarrow$ todas as strings com a e b, com pelo menos duas letras “a” consecutivas

- ▶ **Exercício Rápido**
Todas as strings com **a** e **b**, começando por **b** e não tendo duas letras **a** consecutivas?

Propriedades Algébricas

- ▶ Comutativa: $r|s = s|r$
- ▶ Associativa: $r|(s|t) = (r|s)|t$
- ▶ Concatenação Associativa: $(rs)t = r(st)$
- ▶ Concatenação Distributiva
$$r(s|t) = rs|rt$$
$$(r|s)t = rt|st$$
- ▶ Vazio é identidade da Concatenação
$$\epsilon r = r$$
$$r\epsilon = r$$
- ▶ Relação entre $*$ e ϵ
$$r^* = (r|\epsilon)^*$$
- ▶ $*$ é idempotente: $r^{**} = r^*$
- ▶ Operador $|$ é comutativo e associativo
- ▶ Operador $*$ é potência

Extensões de Expressões Regulares

- Uma ou mais instâncias – operador unário pós-fixado $+$

$$r^* = r^+ | \epsilon$$

$$r^+ = rr^* = rr^*$$

- Zero ou mais instância – operador unário pós-fixado $?$

$$r? = r | \epsilon$$

Precedência equivalente a $*$ e $+$

- Classes de caracteres

$$[A - Z] = A|B|C|D|\dots|X|U|V|W|Z$$

$$[0 - 9] = 0|1|2|3|4|5|6|7|8|9$$

- Outras extensões

- Ver o manual do *grep*

Definições Regulares

- ▶ Nomear certas expressões regulares (*nome*)
- ▶ Usá-las em ERs subsequentes (fazendo parte das *er*)

$nome_1 \rightarrow er_1$

$nome_2 \rightarrow er_2$

...

$nome_n \rightarrow er_n$

- ▶ **Restrições**
 - ▶ Cada $nome_i$ é definido uma vez e não faz parte do alfabeto
 - ▶ Cada er_i tem símbolos do alfabeto e nomes definidos antes
- Evitar definições recursivas
- ▶ Exemplo para os identificadores da linguagem C
 - $letra_ \rightarrow [A - Z a - z _]$
 - $digito \rightarrow [0 - 9]$
 - $identificador \rightarrow letra_ (letra_ | digito)^*$

Casos de Estudo: Definições Regulares

- ▶ Datas com diferentes separadores

- ▶ Solução

data	→	dia separador mes separador ano
dia	→	$[0-3\epsilon][0-9]$
mes	→	$(0 1 \epsilon)[0-2]$
ano	→	aaaa
a	→	$[0-9\epsilon]$
separador	→	$_ . -$

- ▶ Ponto flutuante sem sinal

- ▶ Solução

num	→	digitos fracao expoente
digito	→	$0 1 2 \dots 9$
digitos	→	digito <i>digitos</i> *
fração	→	$.\text{digitos} \epsilon$
expoente	→	$(E(+ - \epsilon)\text{digitos}) \epsilon$

Exercícios

- ▶ Descrever as seguintes linguagens

- ▶ $(a|\epsilon)(b|ba)$

- ▶ $0^*10^*10^*10$

- ▶ $(aa|bb)^*((ab|ba)(aa|bb)^*(ab|ba)(aa|bb)^*)^*$

ER comuns em Compiladores

- Algumas ER comuns em analisadores léxicos

if	→	if
then	→	then
else	→	else
relop	→	< <= = >= >
identificador	→	letra(letra digito)
número	→	digito*(.digito+)?(E(+ -)?digito+)?

- Pergunta

Será que tmp_1 é um identificador?

Diagramas de Transição

- ▶ Possuem
 - ▶ Coleção de nós que representam estados
 - ▶ Arestas entre estados, rotulada por um ou mais símbolos
- ▶ Algumas convenções
 - ▶ Estados de **aceitação** (ou finais)
 - ▶ Estados de **aceitação com ***
 - ▶ Estado **inicial**
- ▶ Exemplo, supondo
identificador \rightarrow letra (letra | dígito)*
- ▶ Exemplo, supondo
relop \rightarrow $< \mid > \mid <= \mid >= \mid = \mid < >$

Definição de Autômato Finito

- ▶ Autômato finito M sobre alfabeto Σ é $(K, \Sigma, \delta, e_0, F)$
 - ▶ K é um conjunto finito de estados
 - ▶ Σ é o alfabeto de símbolos da linguagem
 - ▶ $\delta : K * \Sigma \rightarrow K$ é a função de transição de estados
 - ▶ e_0 é o estado inicial
 - ▶ F é o conjunto de estados finais
- ▶ δ é parcial

Exemplo de Autômato Finito

- ▶ Autômato que reconhece números reais e inteiros

- ▶ Definição

- ▶ $M = (K, \Sigma, \delta, e_0, F)$
 - ▶ $\Sigma = \{d, .\}$
 - ▶ $K = (e_0, e_1, e_2, e_3)$
 - ▶ $F = (e_1, e_3)$
 - ▶ $\delta(e_0, d) = e_1$
 - ▶ $\delta(e_1, d) = e_1$
 - ▶ $\delta(e_1, .) = e_2$
 - ▶ $\delta(e_2, d) = e_3$
 - ▶ $\delta(e_3, d) = e_3$

- ▶ Tabela de Transições

	d	.
e ₀	e ₁	
e ₁	e ₁	e ₂
e ₂	e ₃	
e ₃	e ₃	

Segundo Exemplo de Autômato Finito

		a	b
inicial	q ₀	q ₁	q ₂
	q ₁	q ₀	q ₂
final	q ₂	q ₂	q ₂

- Este autômato corresponde a qual expressão regular?

Dois tipos de autômatos: AFD e AFND

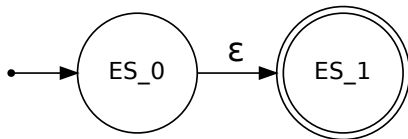
- ▶ **Autômato Finito Não-Determinístico (AFND)**
 - ▶ Tem um conjunto de estados S
 - ▶ Funções de transição
 - ▶ Um estado de partida
 - ▶ Um conjunto de estados finais (de aceitação)
- ▶ **Autômato Finito Determinístico (AFD)**
 - ▶ Como um AFND
 - ▶ Não tem transições vazias
 - ▶ No máximo uma transição de saída por símbolo

Expressão Regular \rightarrow AFND

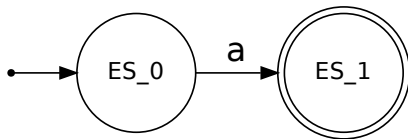
- ▶ Construção de Thompson
 - ▶ Cada ER básica se traduz em um AFND
 - ▶ Pode-se agregar os AFND conforme se agregam as ERs
- ▶ Cada AFND tem exatamente um estado de partida e um estado final

Reconhecedores básicos

- AFND para reconhecer ϵ

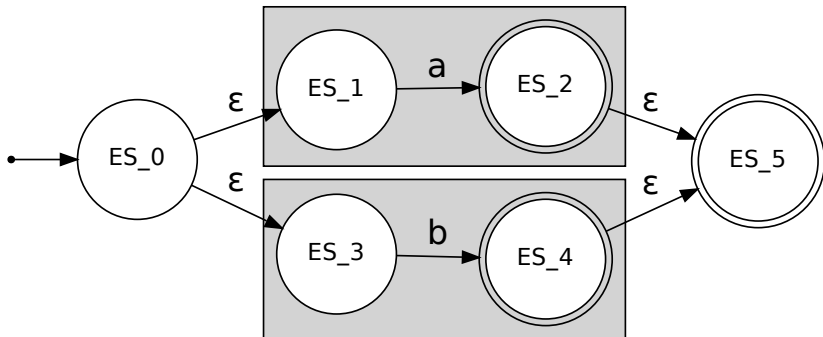


- AFND para reconhecer um símbolo a



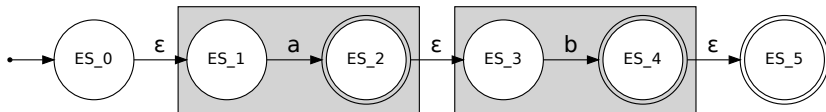
Reconhecedor de Alternativa

- AFND que reconhece a alternativa $a|b$



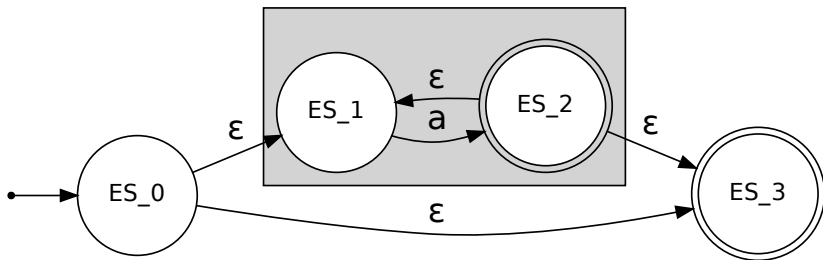
Reconhecedor de Concatenação

- AFND que reconhece a concatenação **ab**



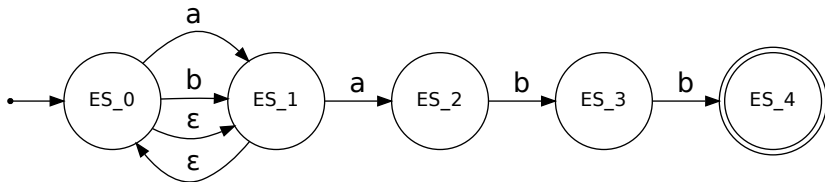
AFND reconhecedor do Fechamento de Kleene

- AFND que reconhece a^*



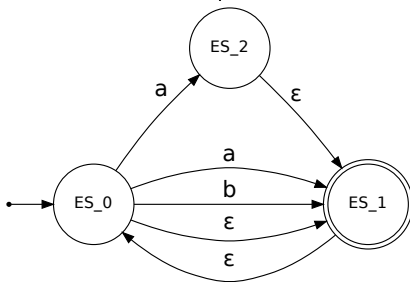
Exercício

- Construir um AFND que reconheça $(a|b)^*abb$
- Solução



Problema dos AFND

- ▶ Autômato Finito Não-Determinístico (AFND)
 - ▶ Bastante poderoso para implementar ERs
 - ▶ Trivial aplicação
 - ▶ Um AFND para cada definição regular
 - ▶ Combinação de todos os AFNDs com ϵ -transições (Estados inicial e final únicos)
- ▶ Problema
 - ▶ ϵ -transições
 - ▶ Várias transições de saída com o mesmo símbolo



- ▶ Fácil para a fase de projeto, difícil de implementar

Conclusão da Aula de Hoje

- ▶ Conversão: AFND \rightarrow AFD
 - ▶ Método automático através da construção de subconjuntos
 - ▶ Assunto da próxima aula
- ▶ Livro do Dragão, capítulo 3
 - ▶ Tokens e uso de Expressões Regulares: 3.3
 - ▶ Autômatos: 3.6
 - ▶ Construção de Thompson: 3.7
- ▶ Série Didática
 - ▶ Compiladores, capítulo 2
 - ▶ Linguagens Formais (Blauth): Capítulo 3
- ▶ Próxima aula
AFND \rightarrow AFD e FLEX