



UNIVERSIDADE EDUARDO MONDLANE
FACULDADE DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA

COMPILADORES

Introdução ao Analisador Léxico

Docentes: Ruben Moisés Manhiça
Cristiliano Maculuve

Maputo, 8 de março de 2024



Conteúdo da Aula

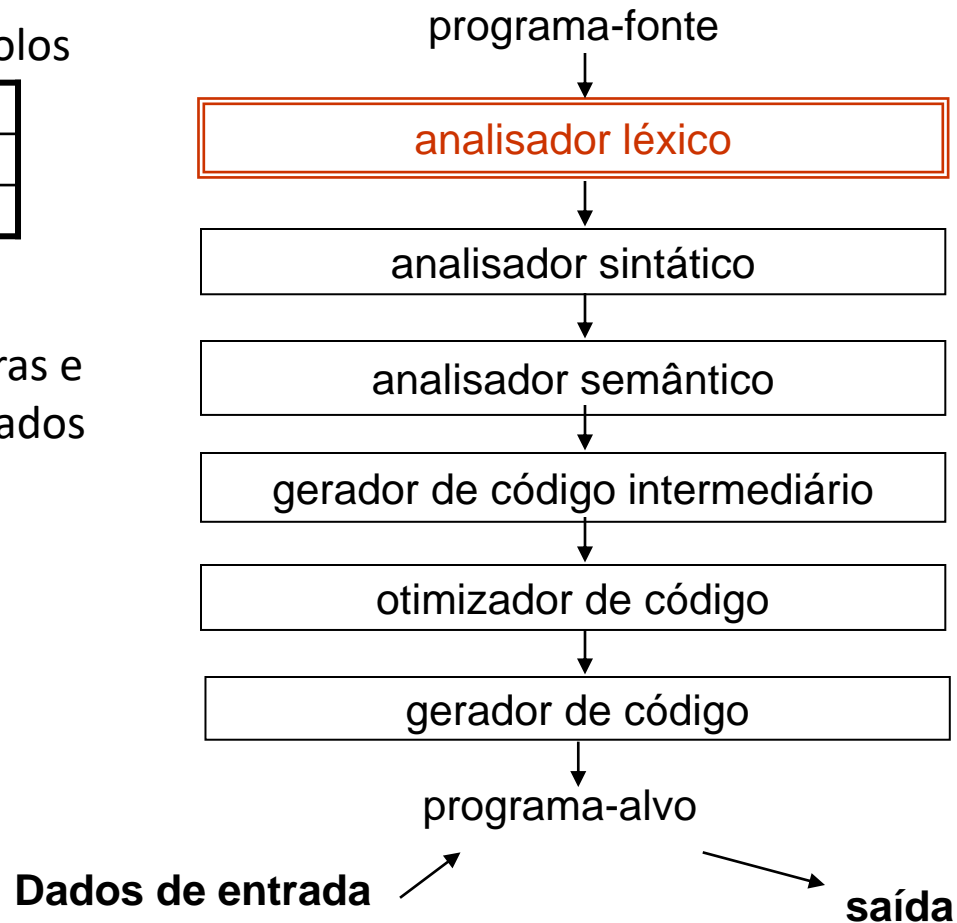
1. Introdução ao Analisador Léxico
2. Introdução a Teoria dos Autómatos



Estrutura geral de um compilador

Tabela de símbolos

Tabela de palavras e
símbolos reservados



Manipulação
de erros





Analizador léxico

- Primeira etapa de um compilador
- Funções:
 - Ler o arquivo com o programa-fonte
 - Identificar os tokens correspondentes
 - “um token se estende até que seja encontrado um caractere que não faça parte dele”
 - Relatar erros léxicos
- Exemplos de tokens:
 - Identificadores
 - Palavras reservadas e símbolos especiais
 - Números

<i>Tokens e lexemas</i>	
Tokens	Lexemas
FOR	for
IF	if
WHILE	while
NÚMERO	1089, 142857, 0.2, 3.14159
IDENTIFICADOR	i, j, contador, nomeAluno
OP_SOMA	+
OP_MAIOR_IGUAL	>=
ABRE_PAR	(





Analizador léxico

- Conjunto de procedimentos que reconhecem cadeias válidas e lhes associam tokens
- Cada vez que é chamado, retorna um par cadeia-token para o analisador sintático
- Consome caracteres irrelevantes: espaços em branco, tabulações, códigos de nova linha, comentários
- Produz mensagens de erro apropriadas quando uma cadeia não é reconhecida por algum autômato
 - Tratamento apropriado na implementação do analisador léxico





Exemplo

- Seja a cadeia $x:=y*2;$

Cadeia	Token
x	t_id
:=	t_atrib
y	t_id
*	t_mult
2	t_num
;	t_ptvg





Exemplo: usando códigos numéricos

- Seja a cadeia $x:=y*2;$

Token	Código
t_id	1
t_num	2
t_mult	3
t_atrib	4
t_ptvg	5

Cadeia	Token
x	1
:=	4
y	1
*	3
2	2
;	5





Exemplo

```
program p;  
var x: integer;  
begin  
    x:=1;  
    while (x<3) do  
        x:=x+1;  
    end.
```

Cadeia	Token
program	t_program
p	t_id
;	t_ptvg
var	t_var
x	t_id
:	t_doispt
integer	t_tipo
;	t_ptvg
begin	t_begin
x	t_id
:=	t_atrib
1	t_num
;	t_ptvg
while	t_while
(t_abrepar

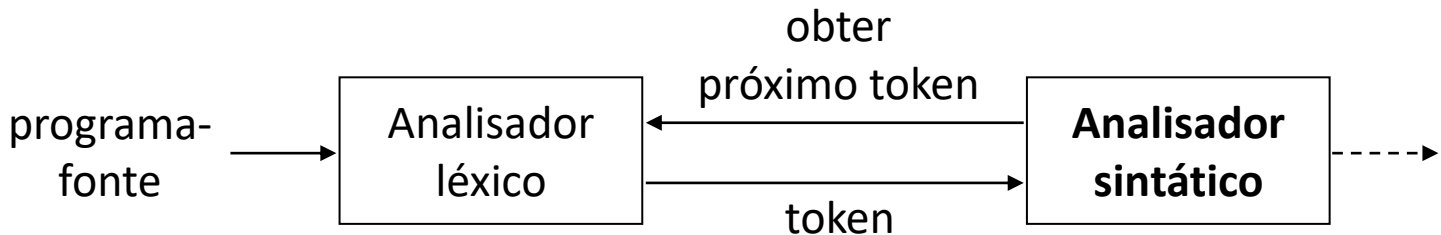
x	t_id
<	t_menor
3	t_num
)	t_fechapar
do	t_do
x	t_id
:=	t_atrib
x	t_id
+	t_mais
1	t_num
;	t_ptvg
end	t_end
.	t_pt





Analizador léxico

- Em geral, subordinado ao analisador sintático
 - Sub-rotina do analisador sintático: a cada chamada, o analisador léxico retorna para o analisador sintático uma cadeia lida e o token correspondente
- O analisador sintático combina os tokens e verifica a boa formação (sintaxe) do programa-fonte usando a gramática da linguagem





Outras funções do analisador léxico

- Consumir comentários e caracteres não imprimíveis (espaço em branco, tabulação, código de nova linha)
- Possível manipulação da tabela de símbolos
- Relacionar as mensagens de erro emitidas pelo compilador com o programa-fonte
 - Deve-se manter contagem do número de linhas
- Diagnóstico e tratamento de erros léxicos





Definições léxicas de uma linguagem

- Exemplos:
 - O que delimita um token?
 - Diferenciação de letras maiúsculas/minúsculas?
 - Qual o conjunto de palavras reservadas?
 - Qual as regras para a formação de identificadores?
 - Quais os operadores aceitos?
 - Quais os delimitadores aceitos? (. ; :) [] { }
 - Quais as regras para a formação de números?
 - Quais as regras para a formação de comentários?
 - etc.





Exemplo

Fragmento de gramática:

```
cmd  →  if expr then cmd  
       / if expr then cmd else cmd  
expr  →  termo relop termo  
       / termo  
termo →  id  
       / num
```

Regras de formação dos itens léxicos:

if	→ if
then	→ then
else	→ else
relop	→ < <= = <> > >=
id	→ letra (letra dígito)*
num	→ dígito+(.dígito+)?(E(+ -)?dígito+)?





Erros léxicos

- Erros
 - Símbolo não pertencente ao conjunto de símbolos terminais da linguagem: @
 - Identificador mal formado: j@, 1a
 - Tamanho do identificador: minha_variável_para_...
 - Número mal formado: 2.a3
 - Tamanho excessivo do número: 5555555555555555
 - Fim de arquivo inesperado (comentário não fechado): {...
 - Char ou string mal formados: 'a, "hello world
 - Os erros detectáveis nessa etapa são limitados
 - Visão local do programa-fonte, sem contexto
- if (a>b) then...





Projeto do analisador léxico

- É desejável que se usem notações formais para especificar e reconhecer a estrutura dos tokens que serão retornados pelo analisador léxico
 - Evitam-se erros
 - Mapeamento mais consistente e direto para o programa de análise léxica
- Notações
 - Gramáticas ou expressões regulares: especificação de tokens
 - Autômatos finitos: reconhecimento de tokens





Expressões regulares

- Determinam conjuntos de cadeias válidas
 - Linguagem
- Exemplos
 - Identificador: letra (letra | dígito)^{*}
 - Número inteiro sem sinal: dígito⁺
 - Número inteiro com sinal: (+ | -) dígito⁺





Autômatos finitos

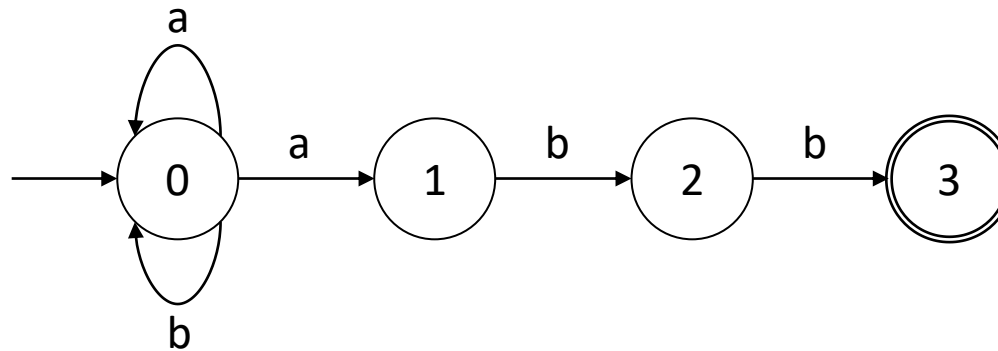
- Modelo matemático
 - Conjunto de estados S
 - Conjunto de símbolos de entrada Σ (alfabeto)
 - Funções de transição que mapeiam um par estado-símbolo de entrada em um novo estado (δ)
 - Um estado inicial s_0
 - Um conjunto de estados finais F para aceitação de cadeias
- Reconhecimento de cadeias válidas
 - Uma cadeia é reconhecida se existe um percurso do estado inicial até um estado final





Exemplo de autômato finito

- $S=\{0,1,2,3\}$, $\Sigma=\{a,b\}$, $s_0=0$, $F=\{3\}$



Quais cadeias esse autômato aceita?

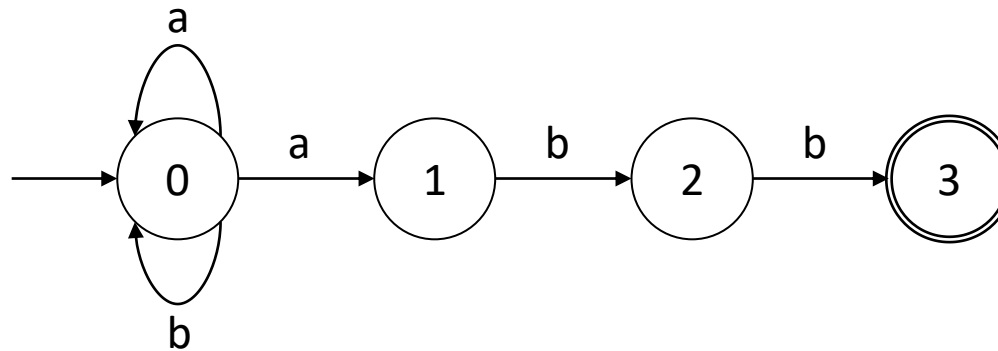
Escreva como uma expressão regular





Exemplo de autômato finito

- $S=\{0,1,2,3\}$, $\Sigma=\{a,b\}$, $s_0=0$, $F=\{3\}$



Quais cadeias esse autômato aceita?

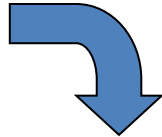
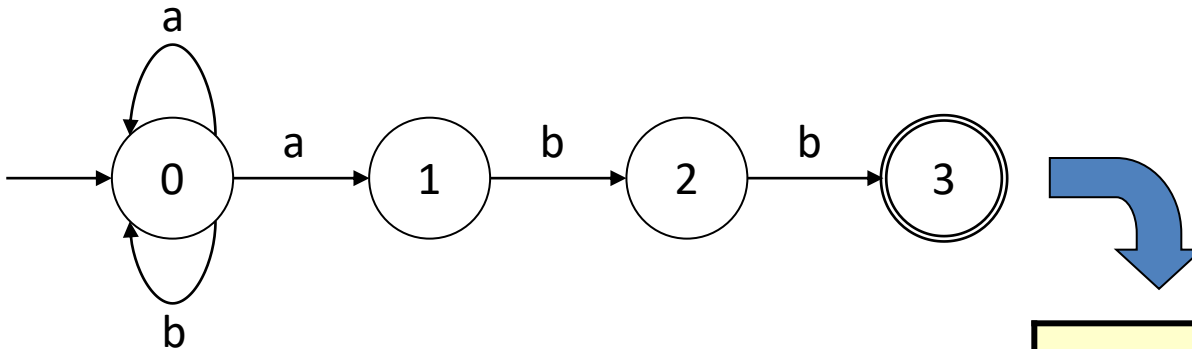
$(a \mid b)^*abb$



Exemplo

Representação em tabela de transição

- Vantagem: acesso rápido
- Desvantagem: pode ocupar grande espaço quando o alfabeto de entrada é grande



Estado	Símbolo de entrada	
	a	b
0	{0,1}	{0}
1	---	{2}
2	---	{3}

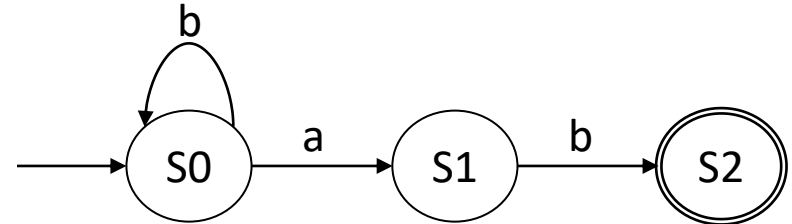


Exemplo de execução do autômato

```
S:=S0
c:=próximo_caractere()
enquanto (c<>eof e s for um estado válido) faça
    s:=transição(s,c)
    c:=próximo_caractere()
fim
se s for um estado final
    então retornar “cadeia aceita”
    senão retornar “falhou”
```

Estado	Símbolo de entrada	
	a	b
S0	{S1}	{S0}
S1	---	{S2}
S2	---	---

$S=\{S_0, S_1, S_2\}$, $\Sigma=\{a, b\}$, $s_0=S_0$, $F=\{S_2\}$



Reconhecer cadeia bab





Execução do autômato

- Opção: incorporação das transições no código do programa
 - Tabela de transição não é mais necessária

$s := s_0$

enquanto (verdadeiro) faça

$c := \text{próximo_caractere}()$

 case (s)

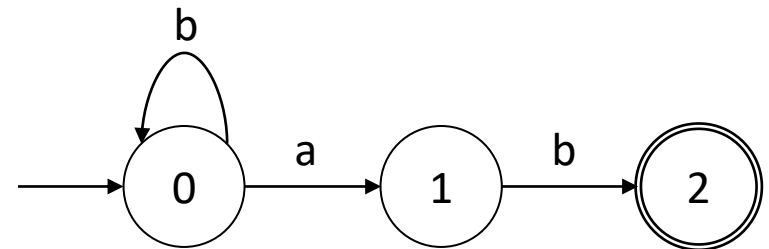
 0: se ($c = 'a'$) então $s := 1$
 senão se ($c = 'b'$) então $s := 0$
 senão retornar “falhou”

 1: se ($c = 'b'$) então $s := 2$
 senão retornar “falhou”

 2: se ($c = \text{eof}$) então retornar “cadeia aceita”
 senão retornar “falhou”

 fim //case

fim //enquanto





Tokens de um programa

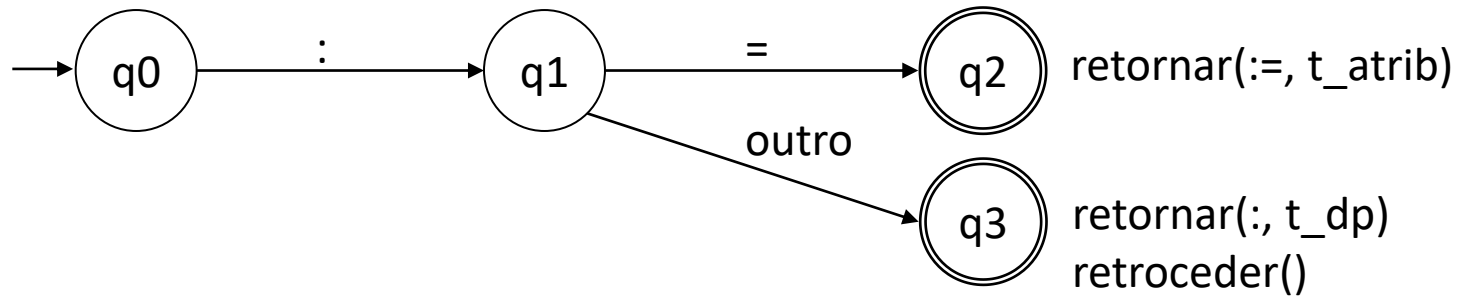
- Exemplos de tokens possíveis
 - Identificadores: `x`, `y`, `minha_variável`, `meu_procedimento`
 - Palavras reservadas e símbolos especiais: `while`, `for`, `:=`, `<>`
 - Números inteiros e reais
- Às vezes, para se decidir por um token, tem-se que se ler um caractere a mais, o qual deve ser devolvido à cadeia de entrada depois
 - Função `retroceder()`





Tokens de um programa

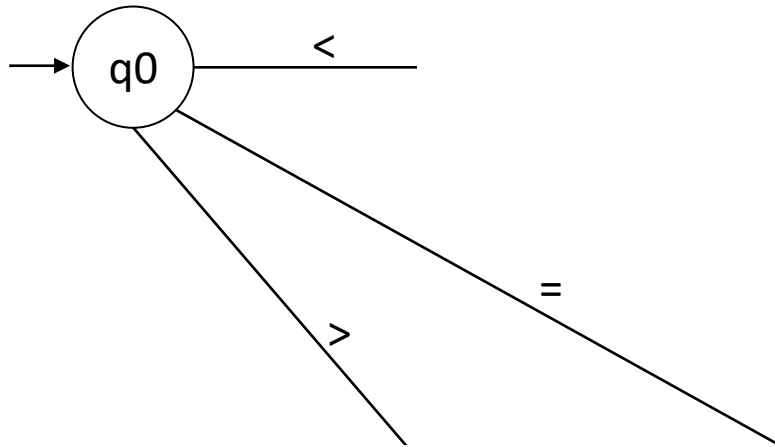
- Autômato para os símbolos `:=` e `:`





Tokens de um programa

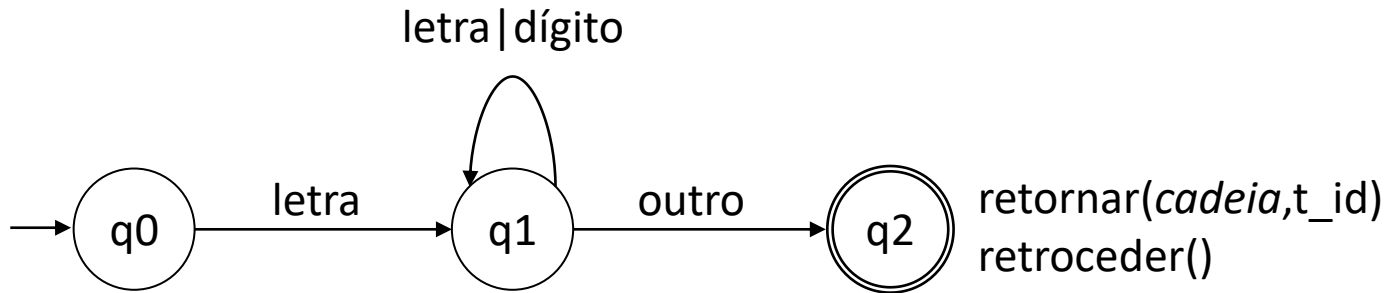
- Exercício: autômato para operadores relacionais $>$, $>=$, $<$, $<=$, $=$ e $<>$





Tokens de um programa

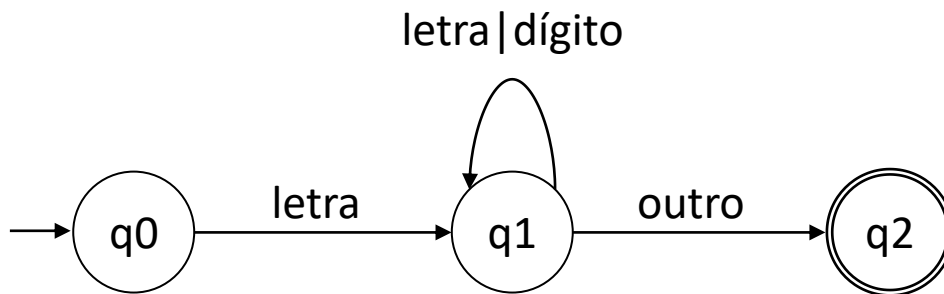
- Autômato para identificadores: letra seguida de qualquer combinação de letras e dígitos





Tokens de um programa

- Autômato para palavras reservadas: while, if, for, array, etc.
- Opções
 - Fazer um autômato para cada palavra-reservada
 - Trabalhoso e ineficiente
 - Deixar que o autômato para identificadores reconheça as palavras reservadas e, ao final, verificar na tabela de palavras reservadas se trata-se de uma palavra reservada
 - Simples e elegante



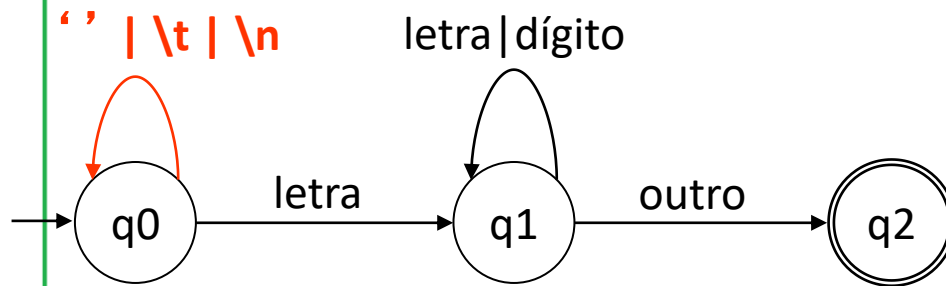
se `busca_tabela(cadeia)=verdadeiro`
então `retornar(cadeia,t_cadeia)`
senão `retornar(cadeia,t_id)`
`retroceder()`





Tokens de um programa

- Autômato para consumir caracteres não imprimíveis: espaços em branco, tabulações e códigos de nova linha
 - O analisador léxico **não** deve produzir tokens para esses símbolos

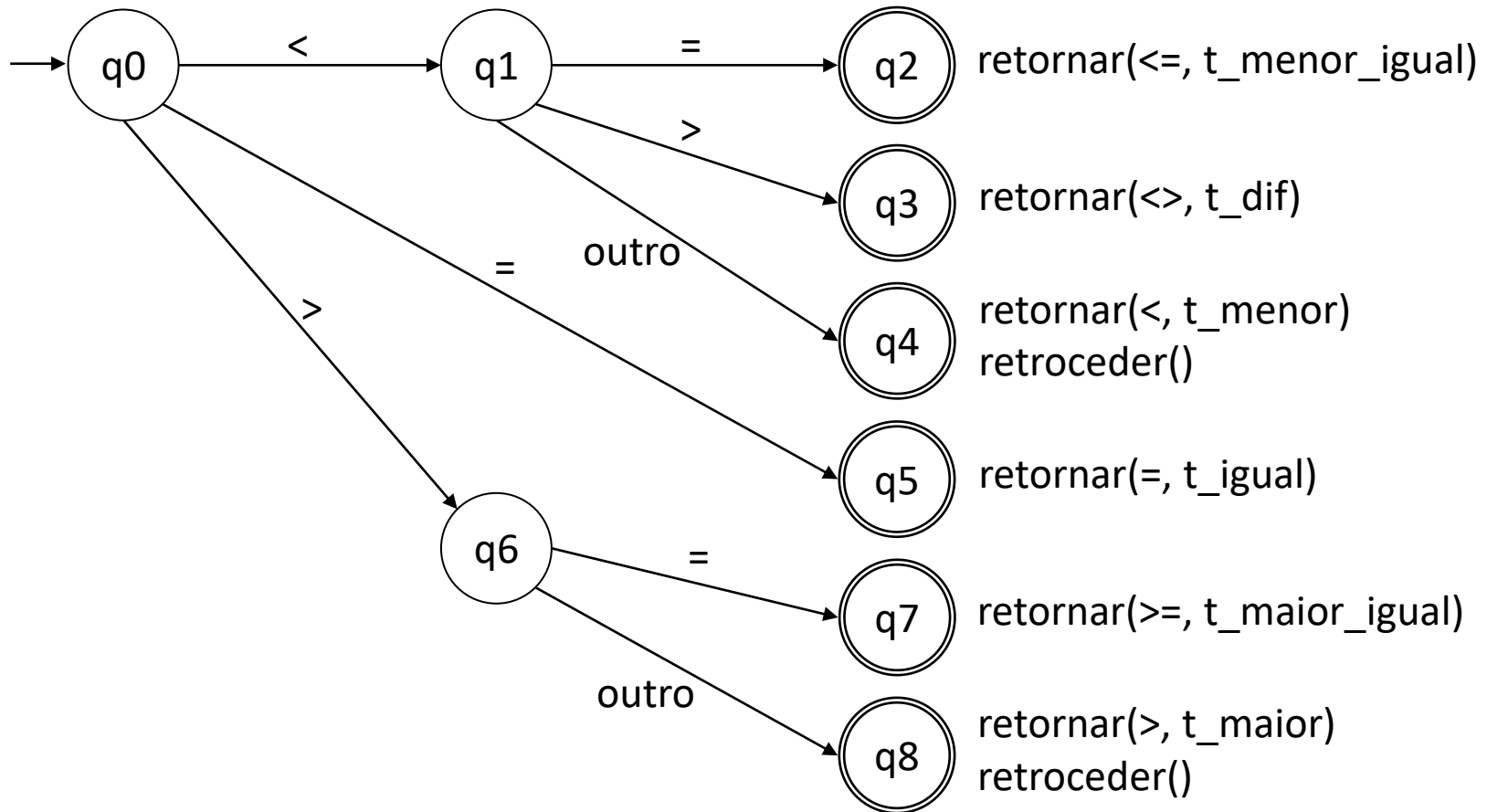


se busca_tabela(*cadeia*)=verdadeiro
então retornar(*cadeia*,t_cadeia)
senão retornar(*cadeia*,t_id)
retroceder()



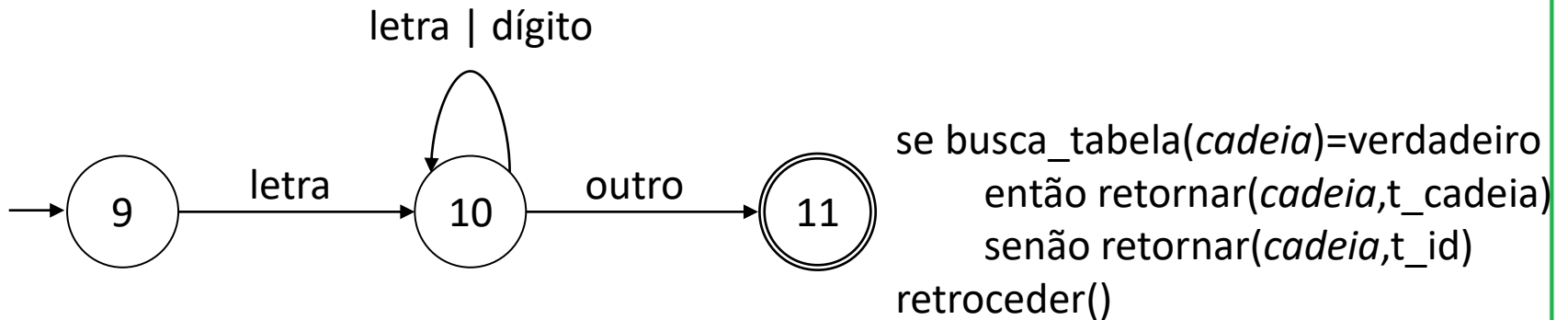


Autômato para o token relop





Autômato para o token id e palavras reservadas





Autômatos para o token num

- A definição é da forma **dígito fração? expoente?**
 - Faremos 3 autômatos:
 - dígito
 - dígito fração
 - dígito fração expoente
- O lexema reconhecido para um dado token precisa ser o mais longo possível
 - O analisador léxico não pode parar após enxergar **12** ou **12.3** quando a entrada for **12.3E4**

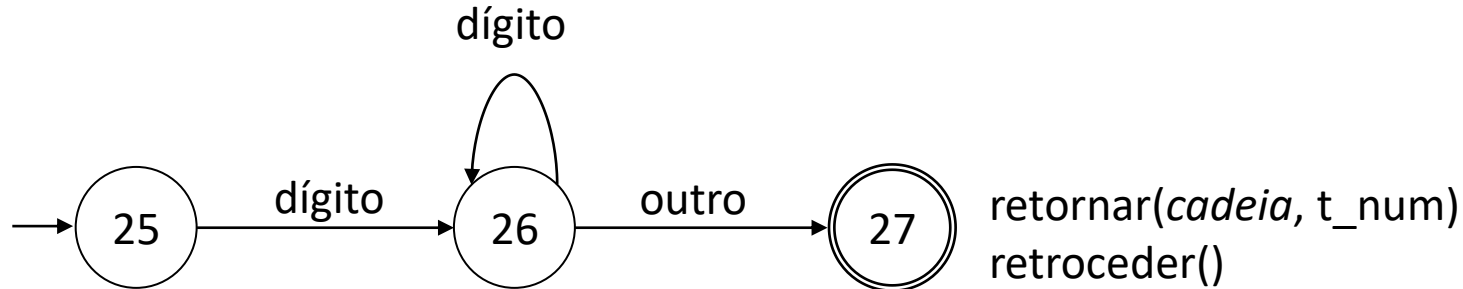




Autômatos para o token num

dígito

Ex. 12

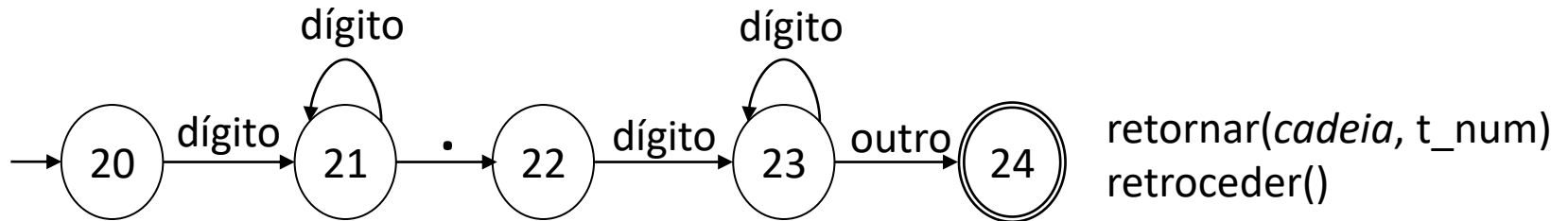




Autômatos para o token num

dígito fração

Ex. 12.3

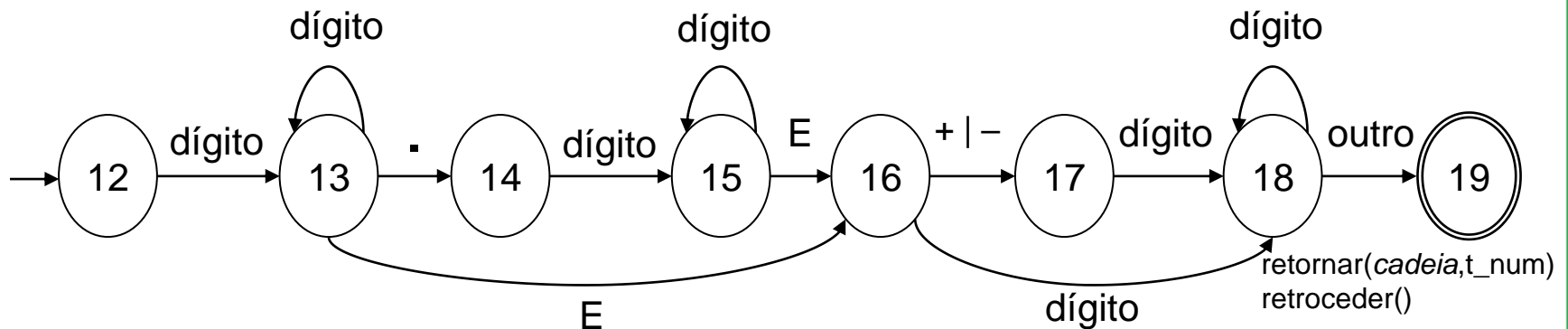




Autômatos para o token num

dígito fração expoente

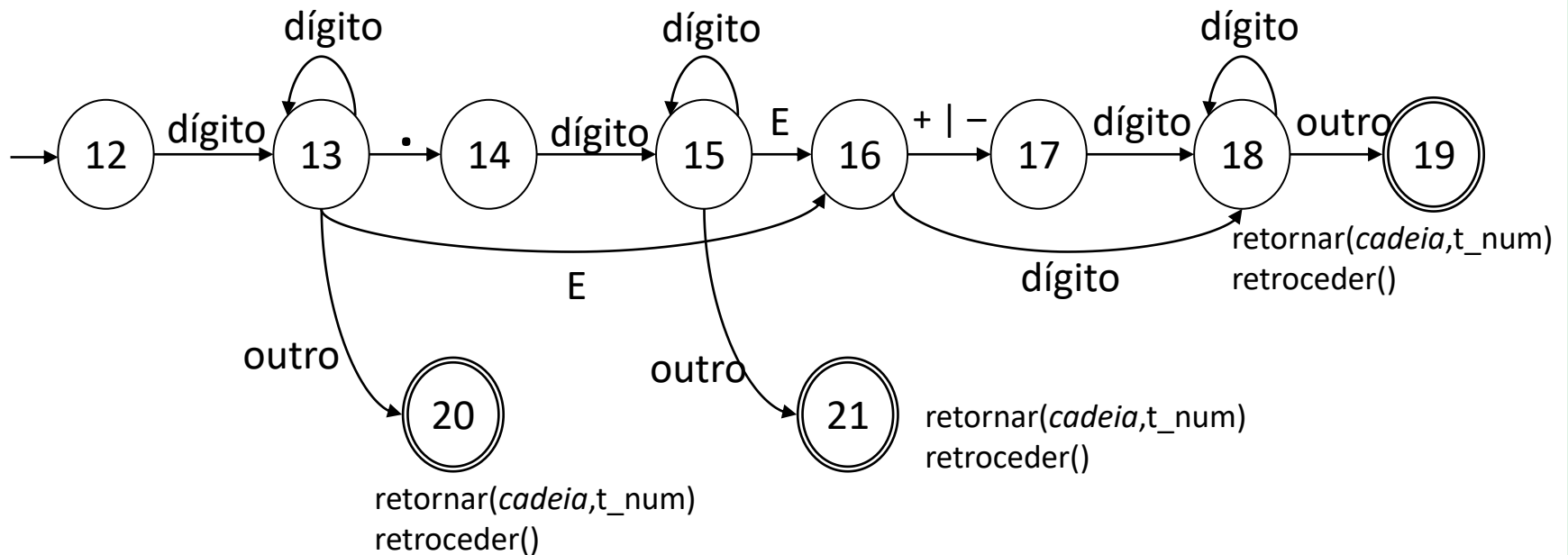
Ex. **12.3E4**





Autômatos para o token num

- Opcionalmente





TPC

Escreva um pequeno programa em java que reconheça o automato do slide 33



FIM!!!

Duvidas e Questões?

