Análise Léxica

Fabio Lubacheski
fabio.lubacheski@mackenzie.br

Análise Léxica

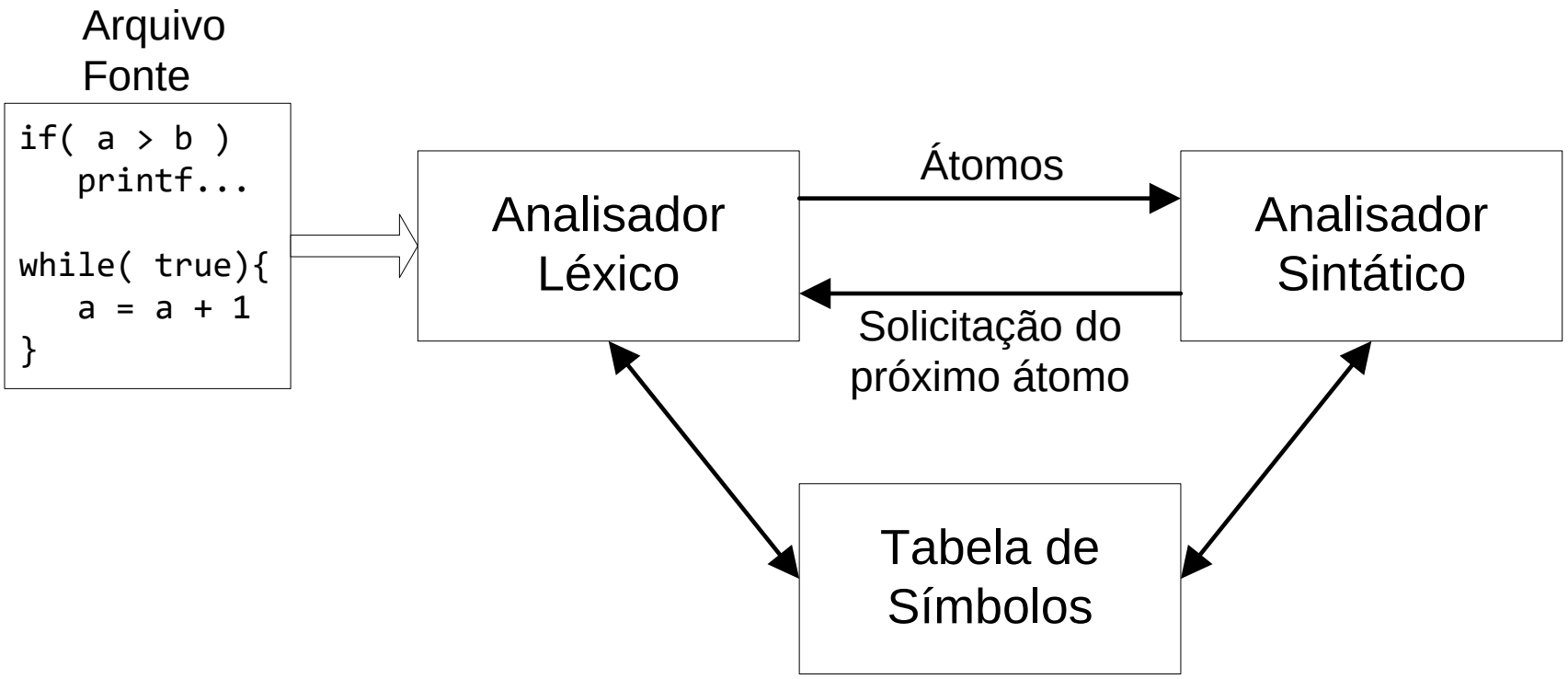
A **análise léxica** lida com técnicas para especificar e implementar um **analisador léxico**.

Um programa-fonte é informado para um **analisador léxico** como sendo uma única cadeia de caracteres. O analisador léxico coleta caracteres e os agrupa logicamente de acordo com um **padrão**, atribuindo um **código interno** aos agrupamentos de acordo com sua estrutura.

Esses **agrupamentos lógicos** são chamados de **lexemas**, e os **códigos internos** esses agrupamentos são chamados de **tokens** *(***átomos***)*

Análise Léxica

Normalmente o analisador léxico é implementado como uma subrotina do analisador sintático.



Funções do Analisador Léxico

- Extração e classificação de átomos
- Eliminação de delimitadores e comentários
- Conversão numérica
- Identificação de palavras reservadas
- Tratamento de identificadores
- Recuperação de erros
- Interação com o sistema de arquivos
- Controle da Numeração de linhas do programa fonte

Átomo

Na maioria das linguagens de programação, as seguintes **agrupamentos lógicos** (**lexemas**) são considerados átomos:

palavras-chaves, operadores, identificadores, constantes, cadeias de caracteres (strings), e símbolos de pontuação como vírgula, ponto e vírgula, dois pontos, e ponto, etc.

Diferentes **lexemas** podem gerar um mesmo **token**, ou seja, um conjunto de **diferentes caracteres** podem determinar o mesmo **átomo**.

Atributos para átomos

Quando mais de uma sequência de caracteres (lexema) casa com um padrão para um átomo, o analisador léxico deve retornar informações adicionais sobre o átomo identificado

> **Exemplo: 10** e **20** podem geram o átomo **NUMERO**, mas é essencial para o gerador de código saber qual cadeia foi de fato reconhecida.

Como descrever a sequência de caracteres (padrão) que podem gerar um determinado **átomo** ?

Expressão regular

O **analisador léxico** entende o programa fonte como uma **linguagem regular**.

Uma **expressão regular** é regra de formação para representar todas as combinações de caracteres que estão associadas a determinado **átomo**.

Considere a expressão regular que descreve o padrão (regra de formação) para números inteiros em uma linguagem de programação.

$(+|-|\varepsilon)(0|1|2|3|4|5|6|7|8|9)^{+}$

ou em outra notação

$(+|-)?(0\text{-}9)^{+}$

Analisador Léxico

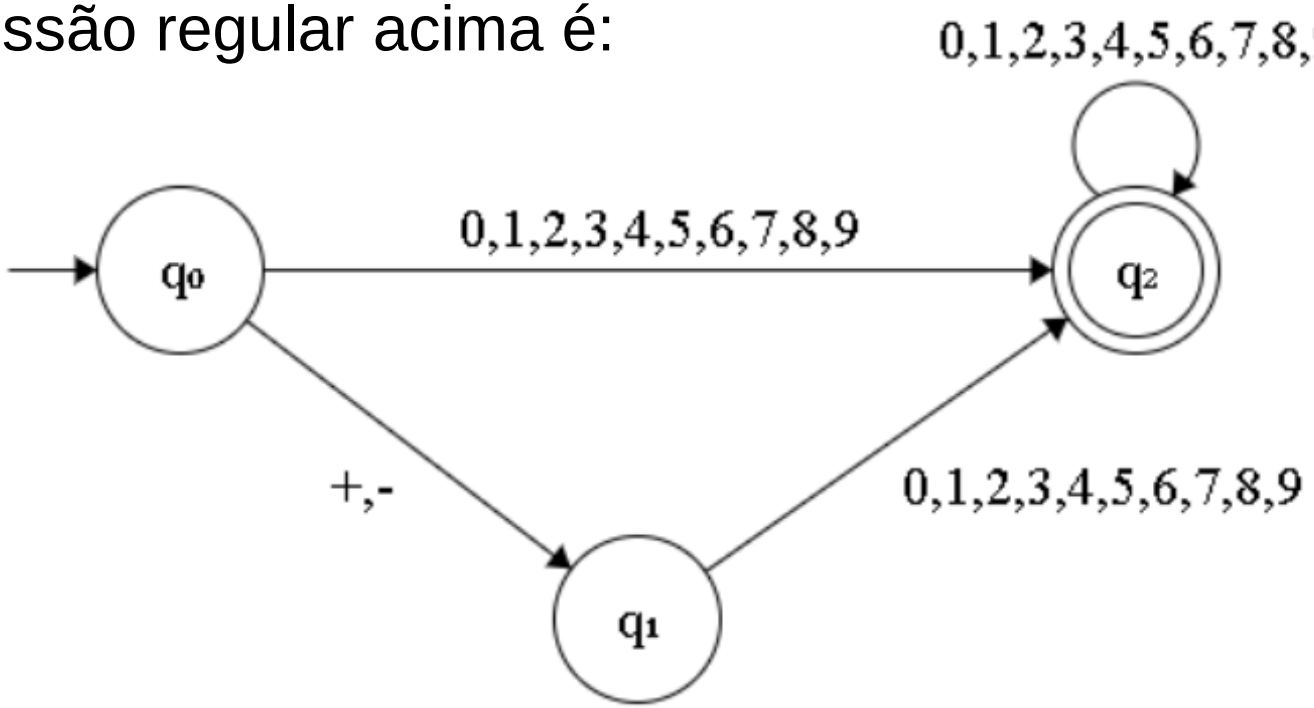
Um **analisador léxico** reconhece os **agrupamentos lógicos** (**lexemas**) de uma arquivo fonte lido de um arquivo texto e produz os átomos para o analisador sintático, a implementação do analisador léxico é baseada em um **autômato finito**.

Um **autômato finito** é uma máquina de estados finitos que reconhece as palavras geradas por uma **expressão regular**.

Implementação do AFD

- Considere a expressão regular vista anteriormente:
  `(+|-|ε)(0|1|3|4|5|6|7|8|9)+`

- O autômato finito que reconhece as palavras geradas pela expressão regular acima é:



Como implementar um AFD

Um **autômato finito** pode ser convertido em **uma função** que toma como entrada cadeias (palavras) e respondem “sim=1” se a cadeia fizer parte da linguagem reconhecida pelo autômato e “não=0” caso contrário.

As cadeias são vetores de caracteres onde sua última posição tem o finalizador de string (‘\x0’) da linguagem C. Dentro da função praticamente usaremos somente rótulos e goto para os rótulos, isso torna a implementação bastante simples.

Vamos implementar a função para o autômato que representa a expressão regular $(+|-|\varepsilon)(0|1|3|4|5|6|7|8|9)^{+}$

Exercícios

1) Considere Σ={a}, apresente a **expressão regular** e o **autômato** para uma linguagem com palavras que possuem o número par de a's. Em seguida, baseado no autômato escreva uma função que implementa o autômato.

2) Considere Σ={a,b}, apresente a **expressão regular** e o **autômato** para uma linguagem com palavras que possuem o número par de a's. Em seguida, baseado no autômato escreva uma função que implementa o autômato.

Exercícios

3) Considere Σ={a,b}, escreva o **autômato** para uma a linguagem com palavras que possuem o número **par de a's** e **ímpar de b's**. Em seguida, baseado no autômato escreva uma função que implementa o autômato.
4) Considere Σ={a,b}, escreva a **expressão regular** e o **autômato** para uma linguagem que **não** o possui dois a's consecutivos. Em seguida, baseado no autômato escreva o analisador léxico equivalente.
5) Reescreva as implementações vistas em sala, trocando as funções que utilizam goto para funções que utilizam `while` e `switch .. case`, ou seja, sem o uso de `goto`.

Exercícios

6) Escreva um programa em C que remova os comentários (de várias linhas `/* */`) de um programa fonte escrito na linguagem C padrão sintaticamente correto.

   O programa fonte original deve ser lido de um arquivo, o seu programa terá como saída um novo arquivo fonte "limpo“, sem os comentários (`/* */`) e tudo que estava dentro dos comentários, e é claro, o programa deverá continuar compilando sem erros como antes.

Resolução do exercício 6

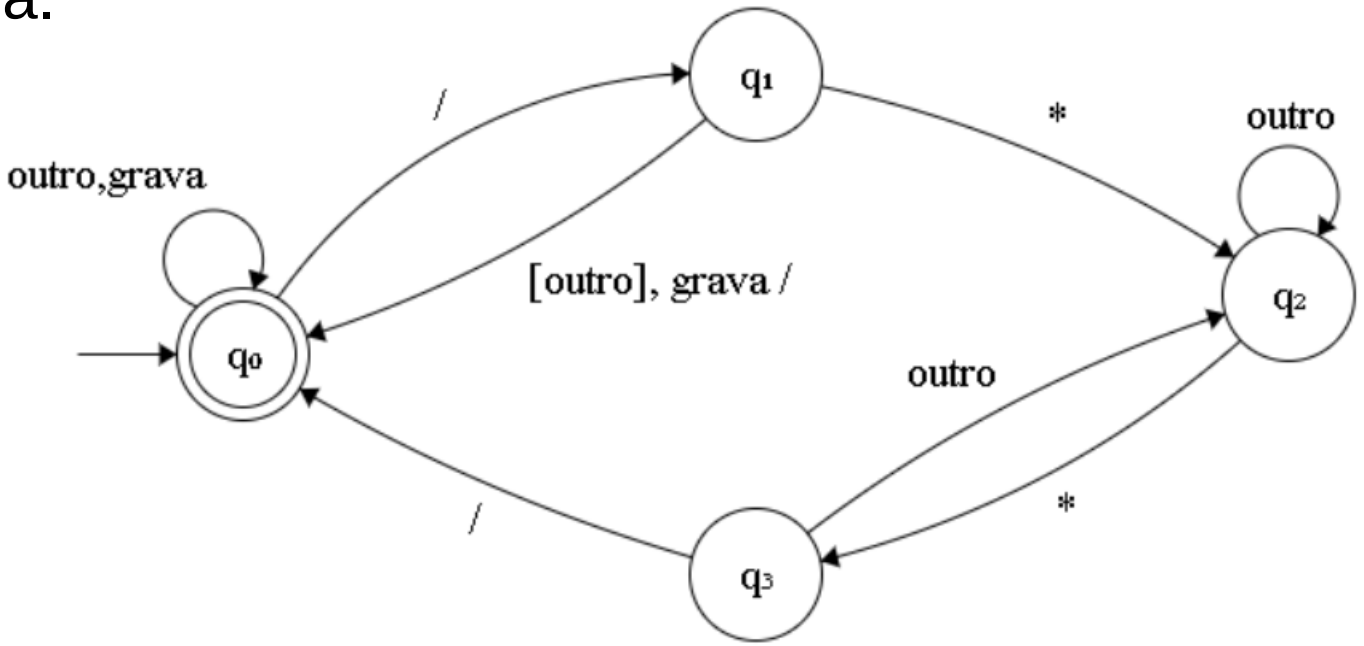
Para eliminar os comentários de um arquivo fonte na linguagem C, podemos usar um diagrama de transições de um AFD, mas não é um AFD no sentido tradicional, pois um AFD precisa ter uma transição para cada estado e caractere, e isso acaba não refletindo o comportamento das funções que reconhecem os átomos de uma linguagem de programação.

No diagrama de transições inserimos algumas informações nas transições que representam ações que serão implementadas na função que representa o diagrama, como a transição **`outro`**, no estado q_0, refere-se a qualquer outro caractere exceto **`'/'`**, além disso o caractere será gravado no arquivo de saída.

Resolução do exercício 6

Na transição **`[outro],grava /`** (q_1 para q_0) os colchetes indicam que caractere identificado deve ser considerado mas não consumindo na transição, isso deverá ser feito no no próximo estado, ou seja, $\mathbf{q_0}$.

Além disso na transição o caractere **`'/'`** é gravado no arquivo de saída.



Resolução do exercício 6

Na implementação da função do diagrama de transição, os caracteres do arquivo fonte serão examinados a partir de um buffer armazenado na memória do computador. Podemos assumir que o arquivo fonte para de entrada estará lexicamente correto, ou seja, todos comentários começados com **“/*”** serão finalizados corretamente com **“*/”**, dessa forma a função sempre encerrará sua execução em q_0.

Fim