Mini Analisador Léxico

Fabio Lubacheski
fabio.lubacheski@mackenzie.br

Mini analisador léxico

Considere uma linguagem de programação cujos os átomos são os seguintes:

- **Identificador**: começado por uma letra e seguido por dígitos ou letras.
- **Número inteiro**: começado digito e seguido mais dígitos;
- **Operador de atribuição**: caractere “`:=`”
- **Palavra reservada while**: caracteres “`while`”

O nosso mini analisador léxico não é sensível ao caso, ou seja, While e while é o mesma palavra reservada.

Os espaços em branco deverão ser ignorados e a **contagem de linha** deve ser feita corretamente.

Definições regulares

- Para facilitar a escrita das expressões regulares dos átomos para o mini analisador léxico vamos dar nome as expressões regulares (**definições regulares**), e assim, a partir das definições regulares mais podemos definir outras definições mais complexas.
- Para o mini alisador léxico teríamos as seguintes **definições regulares**.

  letra $\rightarrow$ A|B|...|Z|a|b|...|z $\approx$ [A-Za-z]

  digito $\rightarrow$ 0|1|...|9 $\approx$ [0-9]

  identificador $\rightarrow$ letra(letra|digito)*

  numero_inteiro $\rightarrow$ digito$^+$

  operador_atribuição $\rightarrow$:=

  while $\rightarrow$ while

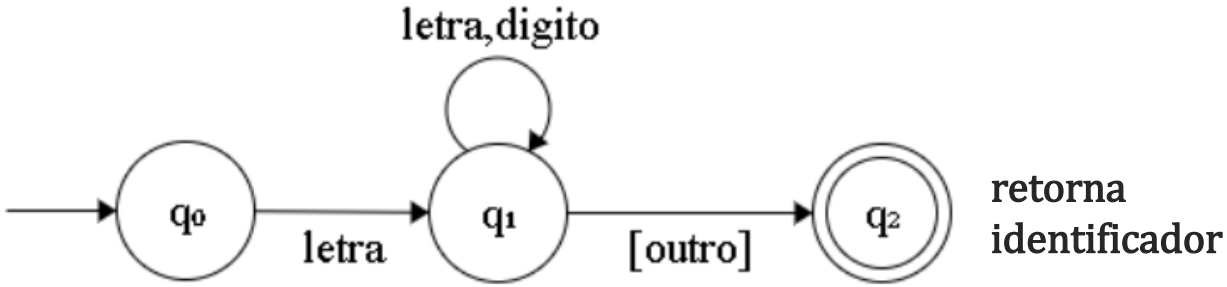
Autômatos para as definições regulares

- Um autômato representa os algoritmos que aceitam cadeias de caracteres que casam com padrões definidos por uma expressão regular.
- Conforme visto na última aula, o diagrama de transições de um autômato não representa todas as ações que um analisador léxico realiza para reconhecer as palavras de uma linguagem.
- Por exemplo não especifica o que ocorre quando temos um erro no reconhecimento da cadeia.
- Por conta disso os autômatos para definições regulares terão algumas modificações em suas transições e estados de **erros**.

Átomo `identificador`

- Para a definição regular para o átomo **`identificador`** temos o seguinte autômato.

  `identificador` $\rightarrow$ `letra(letra|digito)`*



- Lembrando que a transição **`[outro]`**, entre colchetes, indica que caractere identificado deve ser considerado mas não consumindo na transição.

Átomo `while`

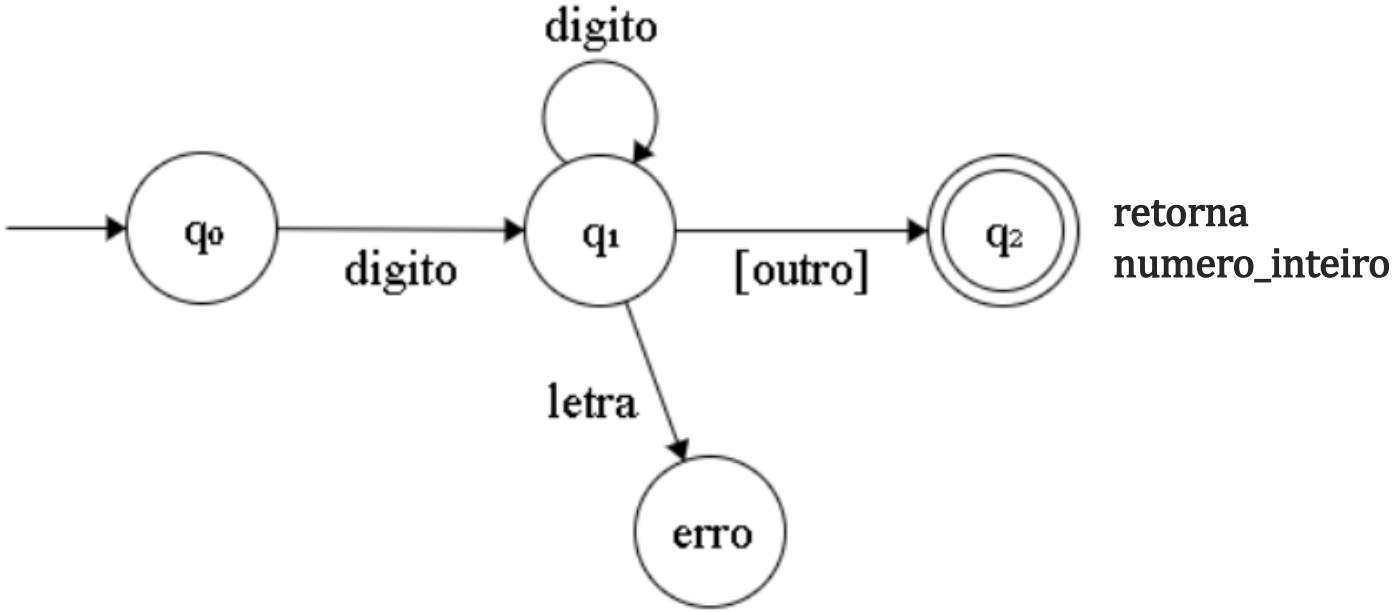
- A rotina que implementa o autômato do átomo **`identificador`** também será usada para reconhecer a palavras reservadas, como por exemplo **`while`**.

- Como as palavras reservadas são sequências de letras, ao invés de fazer um autômato para cada palavra reservada, podemos tratar as palavras reservadas como identificadores especiais.

- Quando a rotina do autômato chegar ao estado final, executamos um código para decidir se o lexema é uma palavra reservada ou um identificador.

Átomo numero_inteiro

- Para o átomo numero_inteiro temos a definição regular **numero_inteiro** → digito$^+$ e seguinte autômato.



- O estado erro é acessado quando após uma sequência de **digito** vier uma letra. Nesse caso o mini analisador léxico para a execução e imprime uma mensagem de erro.

Codificando o átomo

- O processo de análise léxica se assemelha com a atividade de soletrar átomos (constantes inteiras).
- A partir das definições regulares teremos as seguintes constates inteiras simbólicas para os átomos:

```
#define ERRO 0
#define IDENTIFICADOR 1
#define NUMERO_INTEIRO 2
#define ATRIBUICAO 3
#define WHILE 4 // palavra reservada
#define EOS 5 // fim de buffer
```

Codificando o átomo

- Podemos também definir as constantes inteiras como um tipo enumerado.

```
typdef enum{
   ERRO,
   IDENTIFICADOR,
   NUMERO_INTEIRO,
   ATRIBUICAO,
   WHILE,
   EOS
}TAtomo;
```

Rotina principal do mini analisador léxico

- A rotina **`obter_atomo()`** do mini analisador léxico retorna para cada **átomo** reconhecido uma **codificação inteira** (constante) para representar o valor do átomo e o seu **atributo**, caso se faça necessário para o átomo.
- Essas informações serão utilizados pelo analisador sintático avaliar se a sequência de átomos está correta sintaticamente. As informações retornadas pela rotina **`obter_atomo()`** seriam:
  - constante inteira representando o átomo:
  - número da linha onde foi identificado o átomo
  - Os seguintes átomos possuem atributos:
    - **Identificador**: o lexema do átomo, vamos supor que cada lexema terá no máximo 15 caracteres.
    - **Numero inteiro:** o valor inteiro numérico do número

Estrutura para armazenar o átomo

- Para armazenar todas as informações que serão retornadas pela rotina **`obter_atomo()`**, vamos definir um novo tipo utilizando uma estrutura de registros (`struct`) com um conjunto de campos de tipos diferentes para armazenar cada umas das informações referente ao átomo reconhecido:

```
typedef struct{
   TAtomo atomo;
   int linha;
   int atributo_numero;
   char atributo_ID[15];
}TInfoAtomo;
```

Implementação do mini analisador léxico

- Agora podemos desenvolver o código para o mini analisador léxico. O programa deve ler um o arquivo texto e identificar os átomos (tokens) da linguagem.
- A rotina **`obter_atomo()`**, faz a chamada das sub-rotinas que implementam os autômatos vistos, e retorna as informações dos átomos reconhecidos. A declaração da rotina seria:

```
TInfoAtomo obter_atomo(void);
```

- Para simular o analisador sintático você deve fazer uma função com um laço que ficará chamando a função `obter_atomo()` até que chegue ao final do buffer, nesse caso a função `obter_atomo()` retornará o átomo `EOS`.

Implementação do mini analisador léxico

- Além disso, o analisador faz um controle das linhas do programa fonte, e também, a eliminação (ignora) dos delimitadores (espaços em branco, tabulação, nova linha e retorno de carro).
- Para cada átomo reconhecido devem ser apresentados as seguintes informações:

```
{Número da Linha do Átomo, Átomo, Atributo}
```

- Caso ocorra um erro no reconhecimento de um dos átomos do programa fonte analisado, o seu programa deve parar a execução com uma mensagem informando a linha onde ocorreu o **erro.**

Orientações para entrega

- Esta atividade pode ser feita em grupo de até **2 alunos**, não esqueça de colocar o nome dos integrantes do grupo no início do arquivo fonte da implementação.
- Implemente o mini analisador léxico utilizando a estrutura `TInfoAtomo` conforme apresentando na aula.
- O trabalho deve ser implementado na linguagem C e deve estar bem documentado. A entrega do trabalho deve ser feita pelo **Moodle**, você deve enviar o programa fonte e os seus arquivos de testes.
- Como este trabalho pode ser feito **em dupla**, evidentemente você pode “discutir" o problema dado com seus colegas, inclusive as “dicas" para chegar às soluções, mas a dupla deve ser responsável pela solução final e pelo desenvolvimento do seu programa.

Boa sorte !