Aula 004 - Análise Léxica

Autômatos

Prof. Rogério Aparecido Gonçalves
rogerioag@utfpr.edu.br

Universidade Tecnológica Federal do Paraná (UTFPR)
Departamento de Computação (DACOM)
Campo Mourão - Paraná - Brasil

Ciência da Computação

BCC36B - Compiladores

UTFPR

Agenda

1. Processo de Varredura

Processo de Varredura

Especificação x Implementação

- Usamos expressões regulares para dar a *especificação léxica* da linguagem na prática
- Mas como podemos fazer a implementação do analisador léxico a partir desta especificação?
  - Autômatos finitos é a resposta...
  - Algoritmos para converter expressões regulares são conhecidos e podem ser reaproveitados.
  - A implementação de autômatos gera um analisador léxico bastante eficiente

UTFPR

Autômatos Finitos

- Autômatos Finitos ou Máquinas de Estados Finitos são uma forma matemática de descrever tipos particulares de algoritmos (ou "máquinas").
- Podem ser utilizados para descrever o processo de reconhecimento de padrões em cadeias de entrada, e assim podem ser utilizados para construir **sistemas de varredura**.
- Há uma forte relação entre **autômatos finitos** e **expressões regulares**
- Autômatos possibilitam a implementação de uma especificação em Expressões Regulares.

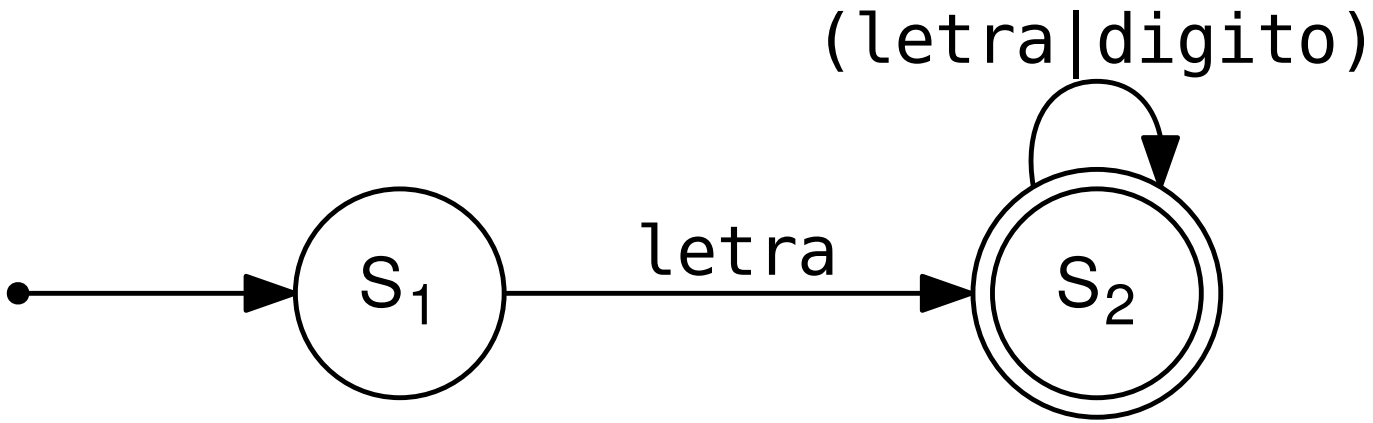
UTFPR

Expressões x Autômatos

- Expressões Regulares para Identificadores

```
letra = [a-zA-Z]+
digito = [0-9]+
identificador = letra(letra|digito)*
```

- Autômato



Expressões x Autômatos

- Expressões Regulares para constantes numéricas

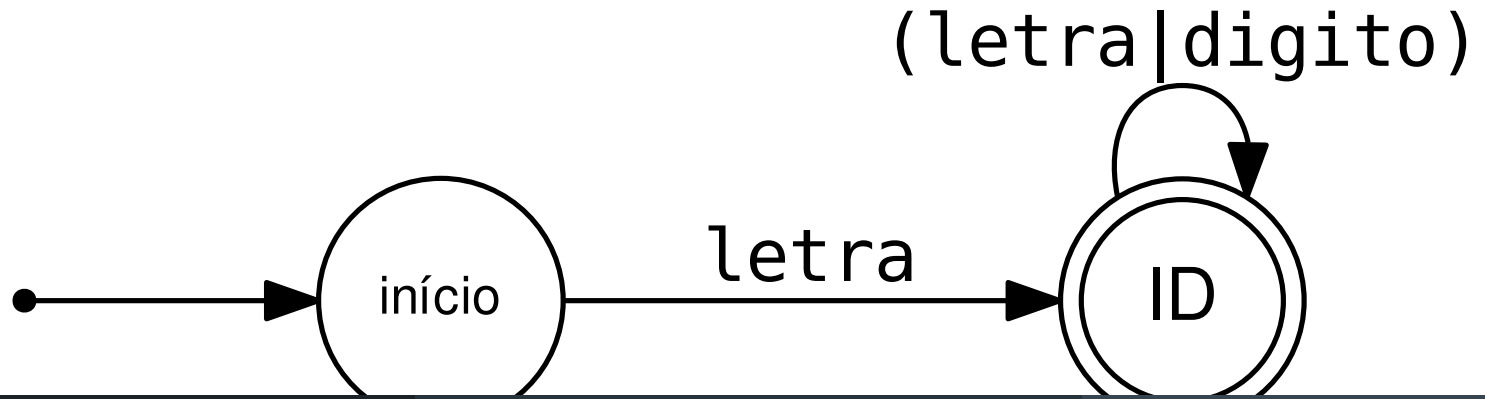
```
digito = [0-9]
natural = digito+
signedNat = (+|-)? natural
numeroPontoFlutuante = signedNat "." natural
numeroNotacao = signedNat("." natural)?(E signedNat)?
```

Autômatos Finitos

- Um autômato finito determinístico (DFA) M é composto por:
  - Um *alfabeto* de entrada: Σ
  - Um conjunto de *estados*: S
  - Uma função de transição entre os estados: $T : S \times \Sigma \rightarrow S$
  - Um estado *inicial*: $S_0 \in S$
  - Um conjunto de estados finais rotulados (estados de aceitação): $A \subset S$
  - A linguagem aceita por M, denotada como $L(M)$ é definida como o conjunto de cadeias de caracteres $c_1 c_2 ... c_n$ no qual cada $c_i \in \Sigma$ e tal que existem estados $s_1 = T(s_0, c_1), s_2 = T(s_1, c_2), ..., s_n = T(s_{n-1}, c_n)$ em que cada s_n é um elemento de A, isto é, é um **estado de aceitação**.

Autômatos Finitos

- Podemos utilizar qualquer sistema de identificação para os estados, inclusive nomes, poderíamos reescrever o diagrama do Autômato para reconhecer identificadores como:
- o nome *letra* representa qualquer letra do alfabeto, segundo sua definição. É uma definição conveniente, pois evita que se tenha que desenhar 52 transições separadas, uma para cada letra em caixa baixa e caixa alta.



Autômatos Finitos

- Temos que $T(início, c)$ está definida somente se c for uma letra e $T(ID, c)$ é definida se c for uma *letra* ou um *dígito*.
- Onde estão as transições que faltam? Elas representam erros.
- Convencionou-se que essas **transições de erros** não sejam desenhadas no diagrama. Assume-se que elas existem.

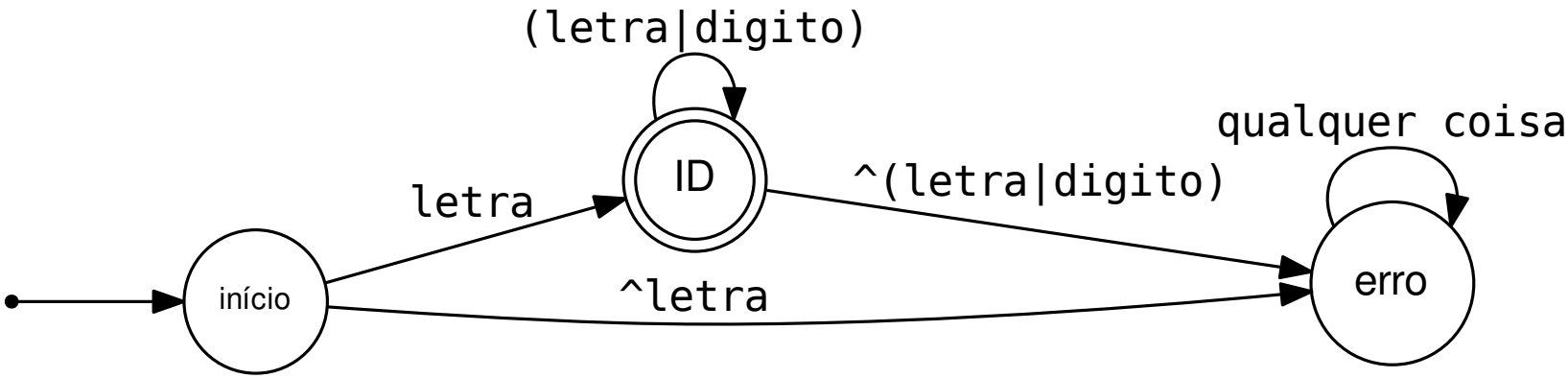


Figura 2: Autômato para Identificadores

Transições vazias

- Uma transição vazia é uma transição que pode ser tomada espontaneamente pelo autômato, sem ler nenhum símbolo da entrada;
- Podemos também construir um autômato que pode tomar mais de uma transição dado um estado e um símbolo;
- Autômatos com transições vazia e múltiplas transições saindo de um mesmo estado para um mesmo caractere são *não-determinísticos*.

DFA vs NFA

- Um DFA é um autômato determinístico, um NFA é não-determinístico;
- Um DFA, dada uma entrada, toma apenas um caminho através dos seus estados;
- Um NFA toma **todos** os caminhos possíveis para uma determinada entrada, e aceita entrada se pelo menos um caminho termina em um estado final.

Autômatos e linguagens

- DFAs, NFAs e REs expressam a mesma classe de conjunto de símbolos:
  - Linguagens regulares.
- Isso quer dizer que podemos converter de um para outro;
- Vantagens específicas:
  - DFAs são mais rápidos para executar
  - NFAs têm representação mais compacta
- Mas... Expressões regulares não são fáceis de entender

UTFPR

Autômatos e linguagens

- DFAs, NFAs e REs expressam a mesma classe de conjunto de símbolos:
  - Linguagens regulares.
- Isso quer dizer que podemos converter de um para outro;
- Vantagens específicas:
  - DFAs são mais rápidos para executar
  - NFAs têm representação mais compacta
- Mas... Expressões regulares não são fáceis de entender - *Por isso usamos expressões regulares para a implementação, e DFAs (ou NFAs) para especificação!*

UTFPR

DFA de análise léxica

- Um DFA de análise léxica tem os estados finais rotulados com tipos de token
- A ideia é executar o autômato até chegar no final da entrada ou levantar um erro caso não seja possível realizar uma transição, mantendo uma pilha de estados visitados e o token que está sendo lido
- Então voltamos atrás, colocando símbolos de volta na entrada, até chegar em um estado final, resultando o tipo do token

Construindo o DFA de análise léxica

- Passo 1: construir um NFA para cada regra, o estado final desse NFA é rotulado com o tipo do token
  - Construção de Thompson
- Passo 2: combinar os NFAs em um NFA com um estado inicial que leva aos estados iniciais do NFA de cada regra via uma transição vazia
- Passo 3: transformar esse NFA em um DFA, estados finais ficam com o rótulo da regra que aparece primeiro
  - Algoritmo de construção de subconjuntos

Criando autômatos para itens da análise léxica (quase na prática)

- Para identificadores
  - Definição de números
  - Definição de letras

Criando autômatos para itens da análise léxica (quase na prática)

- Para identificadores
  - Definição de números
  - Definição de letras
- Números
  - Números inteiros simples
  - Números inteiros com sinal
  - Números reais
  - Números reais em notação científica

Criando autômatos para itens da análise léxica (quase na prática)

- Para identificadores
  - Definição de números
  - Definição de letras
- Números
  - Números inteiros simples
  - Números inteiros com sinal
  - Números reais
  - Números reais em notação científica
- Comentários

Criando autômatos para itens da análise léxica (quase na prática)

- Para identificadores
  - Definição de números
  - Definição de letras
- Números
  - Números inteiros simples
  - Números inteiros com sinal
  - Números reais
  - Números reais em notação científica
- Comentários
- Operadores
  - Aritméticos
  - Lógicos

Implementação de Autômato

Prática

- Com isto, podemos iniciar os trabalhos da disciplina!
- Componentes da linguagem TINY + *invenções do professor* = T++:
- Especificação completa da primeira fase (léxica) disponível no moodle;