Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

Trabalho 01 - Analisador Léxico

SCC0605 Teoria da Computação e Compiladores
Prof. Dr. Thiago A. S. Pardo

Patrick Oliveira Feitosa 10276682
Pedro Henrique Rampim Natali 10310655

São Carlos, São Paulo 02/04/2020

Sumário

Autômatos

Para o desenho dos autômatos, foi utilizado a ferramenta de diagramas do Google Drive, uma vez que ela se mostrou mais amigável além de possuir algumas funcionalidades que o JFlap é desprovido. Em consequência disso, não foi possível fazer o teste dos autômatos

Autômato de início

Esse é o primeiro autômato que a cadeia irá passar. A partir dele, a cadeia poderá ir para os outros autômatos dependendo do seu primeiro caractere. Se o primeiro não levar para nenhum outro ou para o início novamente, o caractere é considerado como erro.

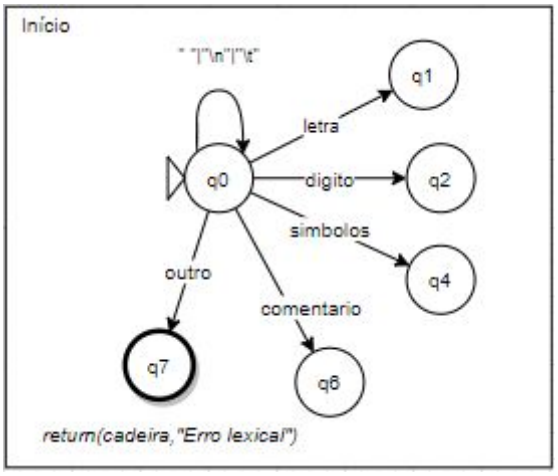

Figura 1: Representação do início do autômato

Autômato de ID/texto

Para entrar ele, basta o primeiro caractere seja uma letra. Ele pode ler dígitos e letras, se aparecer um caractere diferente, retrocede-se e verifica se a cadeia lida é um token e, assim, retornar a cadeia com o seu token, se não, como id.

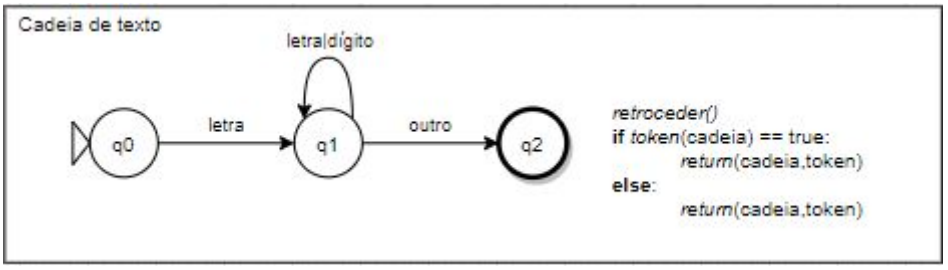

Figura 2: Representação do autômato de leitura de cadeia de texto

Autômato de número

Já o autômato de número pode retornar tanto um número natural, quanto um número real podendo ter o sinal do número. Além disso, foi adicionado um funcionalidade que detecta erro, se não for colocado nenhum número depois do “.”, caso o número for real. Por fim, como o sinais “+” e “-”, podem ser símbolos de soma foi atribuído a esse autômato a detecção desses dois símbolos.

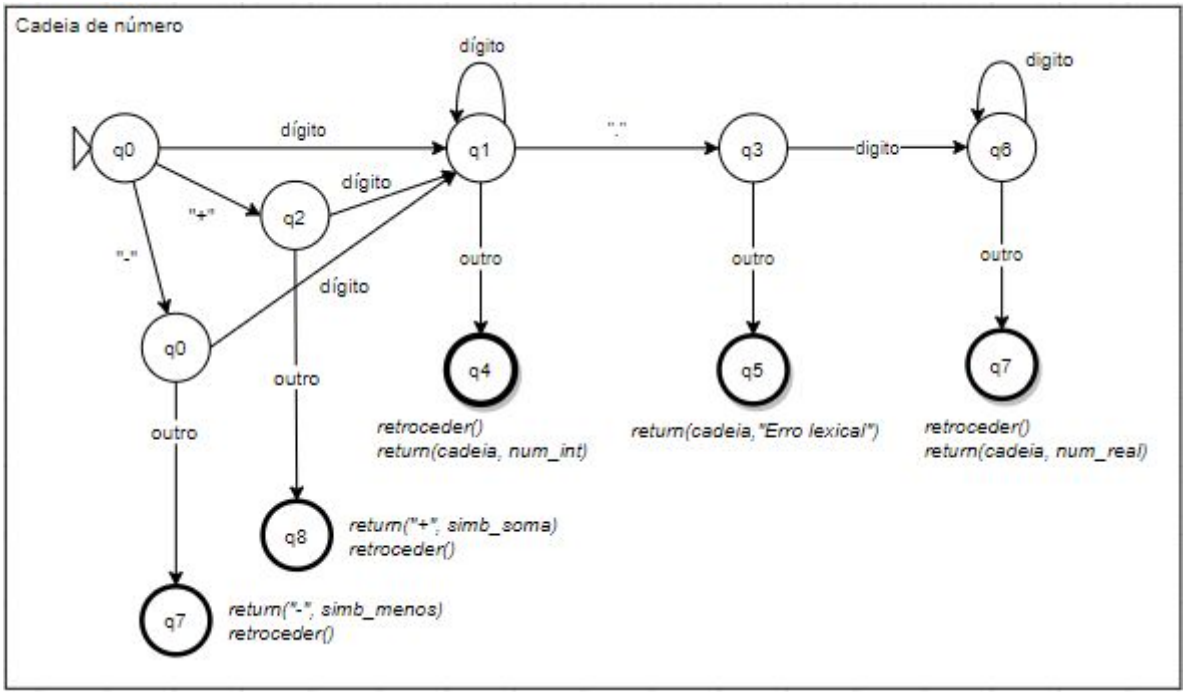

Figura 3: Representação do autômato de leitura de números e dois símbolos.

Autômatos de símbolos

Criamos mais de um autômato para facilitar a visualização, no entanto, poderíamos ter criado apenas um. Ele reconhece os símbolos de comparação, de pontuação, atribuição e de parentes.

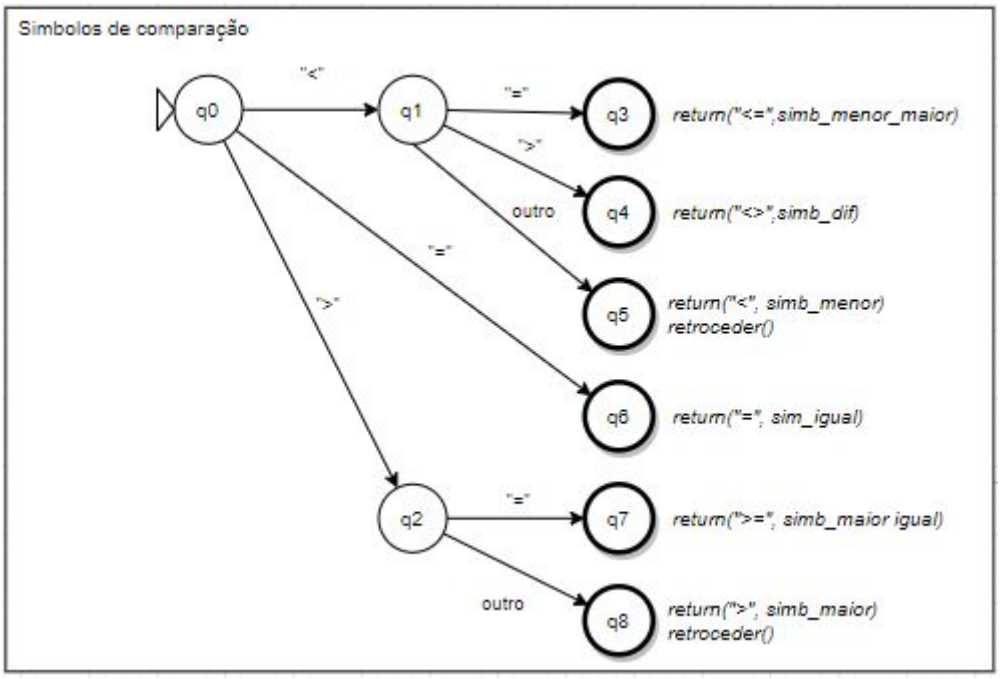

Figura 4: Representação do autômato de leitura de símbolos comparativos.

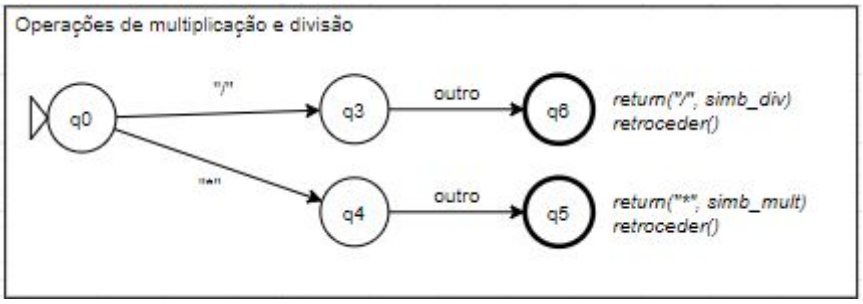

Figura 5: Representação do autômato de leitura de operadores.

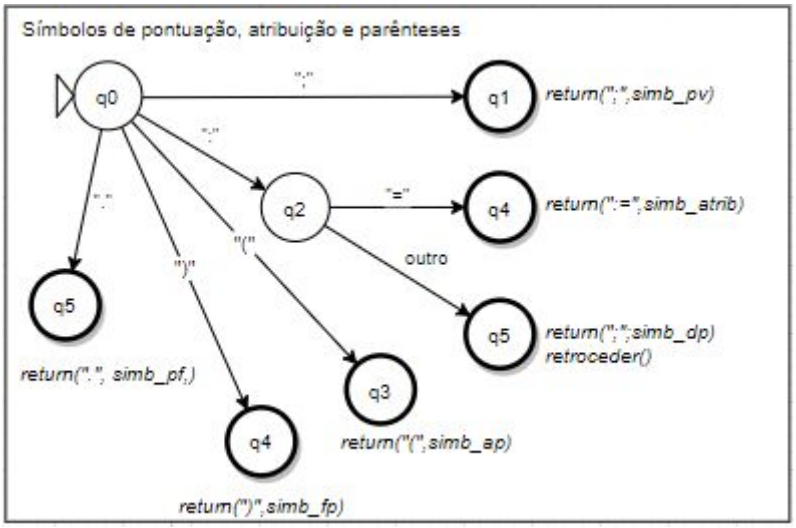

Figura 6: Representação do autômato de símbolos de pontuação, atribuição e parênteses.

Autômato de comentário

Foi criado separado dos demais, pois ele permite que todos os caracteres sejam escritos uma vez que trata-se dos comentário. Para iniciá-lo basta a leitura de “{“ e finalizá-lo com “}”, se o comentário não for finalizado uma mensagem de erro é mandada. Por fim, como os comentários não podem ter mais de uma linha, foi adicionado o reconhecimento desse erro.

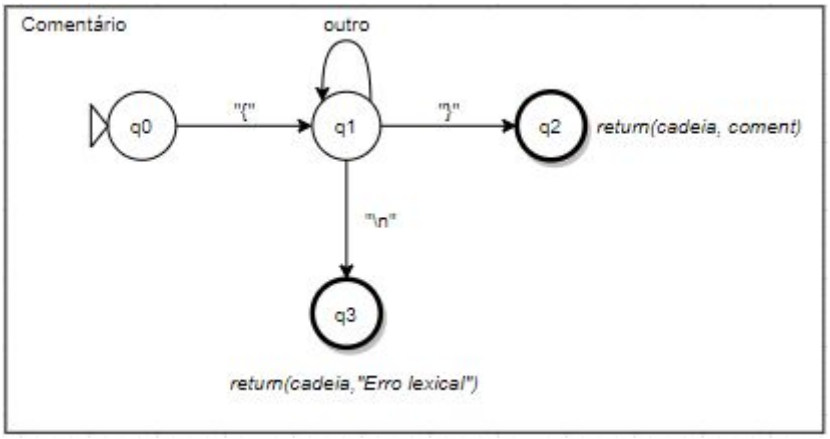

Figura 7: Representação do autômato de leitura de comentários

Decisões de Projeto

Algumas decisões de projeto que foram tomadas a nível de implementação e construção de autômatos vão ser discutidas nesta seção:

- Foram desenvolvidos assim como aconselhado no documento de especificações autômatos para algumas categorias onde o analisador léxico é **iterado várias vezes** para ir delimitando os símbolos a ser catalogado como cadeia/token. Ou seja, ele itera a cada caractere.
- Os autômatos que foram desenvolvidos no código são (comentario, ID, número, comparativo, símbolos e operando). Nota-se que não são exatamente os mesmos desenvolvidos como autômatos visuais. Isso porque em nível de implementação acreditamos que fica mais **organizado e claro** à compreensão dessa maneira, a ponto que com o estímulo visual, é possível utilizar outra organização levemente diferente, sem perder a coerência.
- As tuplas definidas como [cadeia,token] (string/string) são inseridas em uma **tabela de palavra reservada (hash)**, ou seja, um vetor de tuplas. Isso foi feito para que futuramente seja possível aproveitar os dados da tabela caso seja necessário (coisa que não seria possível caso os pares fossem somente impressos no arquivo final sem armazenamento). Nota-se que no caso de cadeias semelhantes, sempre são feitas verificações (busca) para encontrar o token delimitado para a mesma a fim de não haver repetição na tabela e sim no arquivo final.

- Para cadeias especiais que estão presentes na linguagem P, foi definido uma **tabela de palavra reservada especial** onde é possível encontrar o par [cadeia,token] para que, caso encontrados, seja utilizado o token padrão (exemplo: todo begin terá token simb_begin).
- Para **tratamento de erro** algumas políticas foram feitas e serão descritas a seguir:
    - Caso o caractere atual não seja identificado em nenhum autômato descrito, é considerado inválido com resposta “Erro Léxico: caractere [caractere] inválido.”
    - Caso seja inserida algum outro caractere ao montar um número, deve-se reportar o erro e retornar a q0, ou seja, continuar a delimitar os tokens para as seguintes cadeias, reportando o erro como “Erro Léxico: número inválida”.
    - Caso um comentário possua mais de uma linha ou não tenha sido devidamente fechado, é reportado o erro em questão e o iterador voltará para q0.
    - Nota-se portanto que a maneira considerada de avaliação de erro segue o modelo de estado **extra** de erro **individual,** onde há vantagem de maior clareza em comparação com o modelo **genérico.**

Execução

A seguir iremos demonstrar as especificações do sistema onde a aplicação foi desenvolvida e demonstrar um exemplo da mesma para execução.

Especificações

Este programa foi desenvolvido em uma máquina com as seguintes configurações:

Desktop: **Unity 7.4.5** Distro: **Ubuntu 16.04** xenial Dual core **Intel Core i7**-7500U

E é necessário a presença de **Python3** para execução. Caso não seja o caso, execute o seguinte comando abaixo para instalação do mesmo.

sudo apt-get install python3

Para executar o código desenvolvido pelo grupo, é necessário que na pasta do mesmo tenha um arquivo nomeado ‘programa.txt’. Na pasta anexa está presente um exemplo (com resposta esperada para comparação), porém é possível alterar o mesmo que a execução segue normalmente.

No terminal, deve-se entrar no diretório da pasta (exemplo para Linux):

```
cd ~/SCC0605-Compiladores
```

Então deve-se executar o programa da maneira descrita a seguir:

```
python3 main.py
```

O programa então irá executar sobre o documento ‘programa.txt’ e irá devolver uma resposta ‘saida.txt’.

Exemplo de execução

O exemplo a seguir de execução é um alternativo ao anexo junto com o relatório e pode ser executado da maneira descrita anteriormente

program p; {comentario *}* *var x@: integer;* *begin @* *x:=1;* *while (x<>3) do* *x:=20x.1;* *end.*	program, simb_program p, id ;, simb_pv {comentario, Erro lexico: comentario de varias linhas var, simb_var x, id @, Erro lexico: caractere @ invalido em id :, simb_dp integer, simb_tipo ;, simb_pv begin, simb_begin @, Erro lexico: caractere invalido x, id :=, simb_atrib 1, num_int ;, simb_pv while, simb_while (, simb_apar x, id <>, simb_diferente 3, num_int), simb_fpar do, simb_do x, id :=, simb_atrib 20x.1, Erro lexico: numero invalido ;, simb_pv end, simb_end ., simb_p
programa.txt	**saida.txt**