Processamento de Linguagens

Licenciatura em Engenharia de Sistemas Informáticos

2023/24

Trabalho Prático 01

Linguagens Regulares

INTRODUÇÃO

Com este trabalho da unidade curricular de Processamento de Linguagens pretende-se focar a implementação de reconhecedores de linguagens regulares, tendo por base os algoritmos e estruturas de dados abordadas ao longo das aulas. Pretende-se que sejam desenvolvidas implementações em *Python*, conforme exemplificado nas aulas.

OBJETIVOS

Os objetivos do desenvolvimento deste trabalho são os seguintes:

- demonstrar a relevância do uso de expressões regulares;
- definir expressões regulares para o reconhecimento de elementos simples;
- implementar analisadores de expressões regulares baseados em autómatos finitos deterministas
- conhecer o processo de desenvolvimento de ferramentas reconhecedoras de expressões regulares
- implementar os autómatos finitos, adequando a representação conforme se trate de um autómato determinista, ou não determinista
- conhecer o processo de conversão de um autómato não determinista num determinista, especificando funções que realizem a implementação.

ENUNCIADO

As expressões regulares permitem representar de uma forma compacta uma linguagem. O seu reconhecimento pode ser realizado recorrendo a um autómato finito. Neste primeiro trabalho prático pretende-se ter por base a definição formal de uma expressão regular, e a partir dessa representação, suportar a execução do algoritmo de reconhecimento das palavras da linguagem gerada. Sendo o principal objetivo deste trabalho prático, o estudo do algoritmo de reconhecimento de linguagens regulares, não devem ser utilizadas bibliotecas externas de suporte a expressões regulares.

Autómatos Finitos Deterministas

Implementar o algoritmo de reconhecimento de linguagens baseado num autómato finito determinista (AFD). Considerando a representação de um AFD estudado nas aulas deve ser estruturada uma solução para facilitar a utilização desta estrutura de dados, nomeadamente as seguintes funcionalidades:

1. leitura da definição do autómato a partir da sua representação num ficheiro json, validando a sua estrutura, de acordo com as regras de um AFD: arcos com origem e destino num dos elementos do conjunto de estados, o peso deve corresponder a símbolos no alfabeto da linguagem…

Sugere-se que comecem por propor uma representação para um AFD no formato json, tendo por base a representação num dicionário *Python* estudada nas aulas.

2. gera a representação gráfica de um grafo a partir da sua definição, considerando a biblioteca graphviz (gerando o ficheiro de texto que pode ser utilizado como entrada no graphviz).
3. reconhecimento de uma palavra. Além de indicar se a mesma é reconhecida ou não, deverá indicar: - a sequência de estados (caminho que permitiu reconhecer a palavra); - qual foi a situação de erro que permitiu concluir que a palavra não é reconhecida (símbolo não pertence ao alfabeto da linguagem, ou estado que não é final)

Exemplos de utilização:

```
python afd-main.py exemplo.json -graphviz
      digraph {
        node [shape = doublecircle]; q2;
        node [shape = point]; initial;
        node [shape = circle];
        initial->q0;
        q0->q1[label="a"]; q0->q2[label="b"];
        q1->q1[label="b"]; q1->q2[label="a"];
        q2->q1[label="b"]; q2->q2[label="a"];
      }
python afd-main.py exemplo.json -rec ‘aaa’
    ‘aaa’ é reconhecida
    [ caminho q0–a> q1–a> q2–a> q2]

python afd-main.py exemplo.json -rec ‘a!’
    ‘a!’ não é reconhecida
    [símbolo ‘!’ não pertence ao alfabeto]

python afd-main.py exemplo.json -rec ‘ab’
    ‘ab’ não é reconhecida
    [caminho q0–a> q1–b> q1, q1 não é final]
```

De forma semelhante, pretende-se implementar as regras de conversão estudas nas aulas:

- de uma expressão regular para um autómato finito não determinista (AFND)
- de um AFND para um AFD equivalente.

Expressão Regular para AFND

Pretende-se implementar as regras de conversão de uma expressão regular para um autómato finito não determinista (AFND). Relativamente aos AFND deve-se reutilizar as estruturas de dados para o ponto anterior, considerando agora as características específicas do não determinismo.

Uma expressão regular pode ser representada com funções correspondentes aos operadores considerados. Vejamos alguns exemplos:

A expressão regular `a|ab*` deve ser interpretada como a | (a (b)*), pelas regras de prioridade entre os operadores. Associando: uma função **alt** ao operador | de alternativa; uma função **seq** ao operador de concatenação; e uma função **kle** ao operador fecho de kleene, ficamos com seguinte expressão: alt (‘*a*’, seq (*‘a’*, kle (‘*b’*)))

Podemos representar esta expressão em forma de árvore, onde cada nodo tem um operador (*op*) e os respetivos argumentos (*args*), ou é um único símbolo do alfabeto (simb). Seguem dois exemplos:

a | ab* alt (*'a'*, seq (*'a'*, kle (*'b'*))) { "op": "**alt"**, "args": [{ "simb": "a" **},** { "op": "**seq"** , "args": [{ "simb": "a" **},** { "op": "**kle"** , "args": [{ "simb": "b" }] }] }] }	a (€ | b^+) seq (*a*, alt (*epsilon* , trans (*b*))) { "op": "**seq"**, "args": [{ "simb": "a" **},** { "op": "**alt"** , "args": [{ "epsilon": null **},** { "op": "**trans"** , "args": [{ "simb": "b" }] }]}]}

Desenvolva um programa que comece por ler uma expressão regular especificada num ficheiro json de acordo com os exemplos acima, e gere um AFND equivalente (com o formato json estudado no ponto anterior).

Exemplos de utilização:

```
python er-main.py er.json --output afnd.json
```

Conversão de AFND para AFD

Desenvolva um programa que implemente o algoritmo de conversão de um AFND para um AFD, considerando a sua representação no formato json.
Exemplos de utilização:

```
python afnd-main.py afnd.json  -graphviz
  …
python afnd-main.py afnd.json  -output afd.json
python afd-main.py afd.json   -graphviz
   …
```

CRITÉRIOS DE AVALIAÇÃO

São critérios de avaliação do trabalho

1. Qualidade do trabalho produzido
2. Qualidade do Relatório e da Apresentação do trabalho

REGRAS

1. O trabalho tem carácter obrigatório para aprovação à unidade curricular, deve ser realizado em grupo de 3 elementos (no máximo);
2. O trabalho contempla uma **apresentação e defesa individual** em horário a agendar pelo docente. Esta defesa tem aprovação obrigatória, sendo que a falta à defesa corresponderá à não entrega do trabalho pelo aluno (i.e. avaliação de zero valores); Será agendado um horário com cada grupo para a apresentação do trabalho.
3. Não serão aceites entregas ou melhorias após a data definida neste enunciado. Não serão aceites entregas ou melhorias nas épocas de exame (este trabalho apenas é válido para a avaliação da época em que é lançado).
4. O esclarecimento de dúvidas acerca deste documento pode originar a publicação de novas versões.

Bom Trabalho
Óscar Ribeiro & Rui Fernandes