Processamento de Linguagens e Compiladores — MiEFis (3ºano)

Exame de Recurso

Data: 31 de Janeiro de 2019
Hora: 09:00

Dispõe de 2 horas para realizar este exame

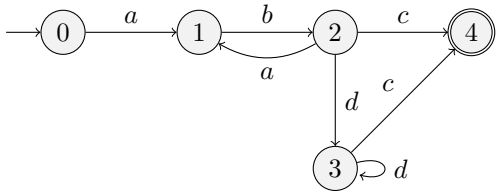
Questão 1: Expressões Regulares e Autómatos (4v)

Responda, então, às seguintes questões:

a) Usando as regras standard de transformação, desenhe um autómato não-determinista correspondente a:

$$c\ (d \mid (ab)^*\ e)$$

b) Construa a expressão regular correspondente ao seguinte autómato e prove a correção da sua transformação mostranndo que as frases geradas a partir do autómato e da expressão regular são as mesmas.



c) Supondo que pretende percorrer um ficheiro de texto `'multas.txt'` com o comando

```
grep -oP expreg multas.txt
```

Escreva uma expressão regular `'expreg'` que permita extrair todas as linhas que tenham matriculas portuguesas.

d) Usando cadeias de derivação (ou autómatos deterministas equivalentes, se preferir), mostre que as duas ERs seguintes (`e1` e `e2`, escritas em notação do FLex) não são equivalentes (não geram a mesma linguagem).

```
e1 = [EeLlEe]
e2 = \[\/?(E|e)(L|l)(E|e)\]
```

Questão 2: Filtros de Texto em Flex e GAWK (4v)

a) Usando o gerador de filtros de texto, baseado em ERs, FLex, escreva filtros distintos para resolver as alíneas seguintes.

- Pretende-se um cifrador de texto que, sempre que encontra um substantivo próprio (sequência de Letras sempre começada por uma maiúscula) o retira da saída e escreve no seu lugar um número inteiro que terá de ser o mesmo sempre que apareça a mesma palavra e diferente para palavras diferentes. Por exemplo, se for dado o texto

  ```
  aqui esta um rapaz Diogo e uma rapariga Ana. no primeiro encontro a Ana disse ao Rui que....
  ```

  sairá assim

  ```
  aqui esta um rapaz 1 e uma rapariga 2. no primeiro encontro a 2 disse ao 3 que....
  ```

- Pretende-se um cifrador de texto diferente que retira o primeiro número que encontra no texto e o usa como offset. Depois sempre que encontra um algarismo o escreve adicionado desse offset (use um incremento circular para voltar ao início quando chega a 9). Por exemplo, se for dado o texto

  ```
  aqui esta uma frase a codificar com base 4: se 1 e menor que 3 e 3 e menor 6....
  ```

  sairá assim

  ```
  aqui esta uma frase a codificar com base : se 5 e menor que 7 e 7 e menor 0....
  ```

b) Observe com atenção a script *GAWK* abaixo apresentada.

```
BEGIN {FS="/"}
$1 == "Ana" && $2 > "40" { print "a" }
$3 > 40                  { print "b" }
$1 ~ /Maria/             { print "c" }
```

e então diga, justificando, o resultado de aplicar essa script ao texto fonte que se segue.

```
Ana Maria Silva/200/20
Ana/200/20
Ana/8/50
Maria Silva/90/200
MariaAna/400/9
```

Questão 3: Desenho/especificação de uma Linguagem (3v)

Pretende-se uma linguagem para descrever um arquivo fotográfico (coleção de uma ou mais fotos) que terá como título o nome do proprietário. Por cada foto deve dar-se um nome (pessoa ou local fotografado), indicar o tipo (preto-branco ou cores), data em que foi tirada (pode não se saber), posição (vertical ou horizontal), nome do fotografo (caso seja conhecido) e uma lista de etiquetas para classificar a foto. Essas etiquetas, são de escolha livre mas terá de existir pelo menos 1 (por exemplo: pessoa, grupo/individual, paisagem, monumento, etc.).

Escreva então, usando a notação do Yacc, uma Gramática Independente de Contexto, GIC, que especifique a Linguagem pretendida.

Questão 4: Gramáticas, Parsing e Tradução (9v)

Considere a seguinte gramática que define uma linguagem para descrever as prendas trocadas num jantar de Natal entre vários amigos (os símbolos terminais estão escritos em minúsculas ou entre apostrofes).

```
p01: Prg      →  Func
p02:          |  Prg ';' Func
p03: Func     →  Ident
p04:          |  Functor
p05: Ident    → id
p06:          | num
p07: Functor  → fid '(' Args ')'
p08: Args     → &
p09:          | Args Func
```

Neste contexto e após analisar a *GIC* dada, responda às alíneas seguintes.

a) Sabendo que um identificador (símbolo `id`) é formado por letras maisculas ou minisculas, digitos, hífens ou underscores desde que começado por uma minúscula, que um número (símbolo `num`) é formado por digitos podendo começar por um sinal opcional e pode conter um ponto a separar a parte inteira e decimal, e que um identificador de função (símbolo `fid`) é um identificador começado por #,
Especifique em Flex um Analisador Léxico para reconhecer todos os símbolos terminais da sua linguagem e devolver os respetivos códigos.

b) Usando a construção de uma Árvore de Derivação, verifique se a frase abaiixo é um programa válido de acordo com a linguagem dada.

```
#f1( #mul( 12 x_1) y_2)
```

c) Desenhe o estado inicial do autómato LR0 da gramática extendida e os estados que dele derivam.

d) Diga porque é que esta gramática não é LL1 e escreva uma que, definindo exatamente a mesma linguagem, seja LL1.

e) Usando a estratégia de parsing recursivo descendente, escreva as funções para reconhecer os símbolos `Functor` e `Args`.

f) Se quiser implementar um parser Top-Down iterativo e table-driven, diga qual deverá ser o conteúdo da stack de parsing LL(1) ao encontrar o fim de ficheiro '$' para que o processo termine em reconhecimento.

g) Usando a notação do Yacc, estenda a GIC com ações semânticas para calcular o número de funções não-identidade que compõem o programa.

h) Usando a notação do Yacc, estenda a GIC com ações semânticas para converter esta escrita *prefix* em *postfix* escrevendo os argumentos um por linha e o nome da função no fim, em outra linha.

i) Usando a notação do Yacc, estenda a GIC com ações semânticas para criar no fim uma lista alfabética de todos os identificadores usados, sem repetições.