Processamento de Linguagens e Compiladores — LCC (3ºano)

1º Teste

Data: 13 de Janeiro de 2017
Hora: 09:00

Dispõe de 2 horas para realizar este exame

Questão 1: Expressões Regulares e Autómatos (4v)

Responda às seguintes alíneas:

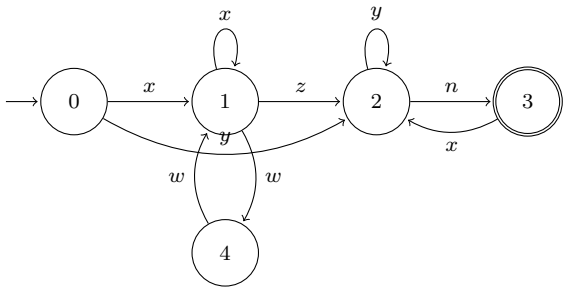
a) Considere as linguagens $L1$ e $L2$ definidas pelas seguintes Expressões Regulares:

$$e1 \;=\; a^{+}c \mid c\; b^{+}$$

$$e2 \;=\; a^{+}c\; b^{+}$$

Usando cadeias de derivação para construir frases de cada linguagem, mostre que $L1$ e $L2$ não são equivalentes.

b) Qual a expressão regular correspondente ao seguinte autómato:



c) Diga, justificando apropriadamente, se as expressões regulares abaixo, escritas em notação do Flex, são equivalentes:

$$(\; a \mid b \;)^{+}\; (\; 1 \mid 2^{+} \mid 3 \;)$$

$$[\; ab \;]^{+}\; [\; 123 \;]^{+}$$

d) Desenhe um autómato determinístico correspondente a:

$$(ba)^{+}\; c^{+}\; (d \mid ab)^{*}\; e$$

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

Especifique filtros de texto com base em expressões regulares e regras de produção (padrão–ação) para resolver as seguintes alíneas:

a) Considere que um sociólogo pretende analisar uma vasta coleção de *pagelas religiosas* (vulgarmente chamadas de "*santinhos*")—tão em voga antigamente para marcar eventos religiosos de **tipos** tais como batisado (Bat), 1ª comunhão (1Com), comunhão solene (SCom), casamento (Casa) e funeral (Fun)—para perceber os períodos de maior uso, os eventos mais marcados, se eram oferecidos pelo festejado para assinalar o momento (OferPor), ou oferecidos pelos convidados ao festejado (OferA), ou o local desses eventos.
Para o efeito, o sociólogo registou a informação de cada *pagela* num ficheiro de texto, um registo por linha, usando o seguinte esquema:

```
identicador!ano:mes!tipoEvento!tipoOferta:nomeFestejado!templo/cidade!fichFoto
```

sendo que o último campo é opcional (pode aparecer ou não).
Por exemplo:

```
s1!1932:maio!1Com!OferA:Serafim Dias!igreja das antas/porto!"fff.jpg"
s2!1932:maio!SCom!OferPor:Maria Rita!igreja do carmo/braga
ss100!1934:agosto!Bat!OferA:Ezequiel Campos!igreja das antas/porto!"ffa.jpg"
```

Escreva então uma script Gawk para:

- calcular o número de santinhos registados em cada ano e em cada mês
- contar os santinhos oferecidos à Maria Rita e identificar os eventos religiosos que originaram essas ofertas
- identificar a cidade onde foram oferecidos mais santinhos por o festejado.

b) Escreva um filtro usando o Flex para ler um texto anotado em XML e transferir esse texto para a saída capitalizando todos os nomes de elementos (tags) usados nas marcas.
Além disso, se as marcas de abertura contiverem atributos depois do nome do elemento, esses pares *atributo-valor* devem ser retirados; pretendendo-se que no fim indique quantas dessas situações oocorreram.

Questão 3: Desenho/especificação de uma Linguagem (4v)

Pretende-se uma linguagem de Domínio Específico que permita descrever um processo de partilhas. Para tal deve-se descrever inicialmente o lote de bens a partilhar; cada objeto tem um identificador próprio, um tipo (ou classe de bem a que pertence), uma descrição e um valor em euros. Depois descrevem-se os herdeiros envolvidos, sendo que cada um tem um código, nome, parentesco com o proprietário desse lote, e email. Finalmente (o objetivo principal da linguagem) é indicar, por cada herdeiro (identificado pelo código), a lista de bens escolhidos (referidos pelo respetivo identificador). Por cada objeto da lista deve ser indicada a ordem de preferência (entre 1 e 3).
Escreva então uma Gramática Independente de Contexto, *GIC*, que especifique a Linguagem pretendida (note que o estilo da linguagem (mais ou menos verbosa) e o seu desenho são da sua responsabilidade).

Especifique em Flex um Analisador Léxico para reconhecer todos os símbolos terminais da sua linguagem e devolver os respetivos códigos.

Questão 4: Gramáticas e Parsing Top-Down (3v)

O parser RD para uma dada linguagem L é mostrado na listagem abaixo.
A partir desse parser, e sabendo que os símbolos terminais são representados por letras minúsculas e os não-terminais são representados por palavras começadas por uma maiúscula, escreva a respetiva gramática independente de contexto (GIC).
Construa a respetiva tabela LL(1).

```
RecL(s: simbT)
  INICIO
    recTerm(i,s); recEs(s); recTerm(f,s)
  FIM
RecEs(s: simbT)
  INICIO
    recE(s); recC(s)
  FIM
RecE(s: simbT)
  INICIO
    SE (s==id) ENTAO recTerm(id,s); recTerm(a,s); recNs(s); recTerm(b,s)
    SENAO erro()
  FIM
RecNs(s: simbT)
  INICIO
    SE (s==b) ENTAO ;
    SENAO SE (s==n) ENTAO recTerm(n,s); recNs(s)
          SENAO erro()
  FIM
RecC(s: simbT)
  INICIO
    SE (s==f) ENTAO ;
    SENAO SE (s==v) ENTAO recTerm(v,s); recEs(s)
          SENAO erro()
  FIM
RecTerm(t: simbT,s: simbT)
  INICIO
    SE (s==t) ENTAO s = daSimb() //analisador léxico
    SENAO erro()
  FIM
```

Questão 5: Gramáticas, Tradução e Parsing Bottom-Up (5v)

Considere a gramática independente de contexto, GIC, abaixo apresentada, que permite declarar uma ou mais variáveis definindo o seu tipo e permite executar isntruções de dois tipos sobre essas variáveis. Note ainda que os símbolos terminais T e não-terminais NT estão definidos antes do conjunto de produções P, sendo S o seu axioma (ou símbolo inicial).

```
T  = { '.', ';', ':', id, i1, i2 }
NT = { S, Ds, Is, As, I, R, Tip, Var }
P = {
p1: S  -> Ds ':' Is
p2: Ds -> &
p3:     | Tip Var As
p4: As -> ';' Tip Var As
p5:     | &
p6: Is -> I R
p7: I  -> i1 Var
p8:     | i2 Var
p9: R  -> '.'
p10:    | ';' Is
p11: Tip -> id
p12: Var -> id
}
```

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

a) Verifique se a frase `Ta varA; Tb varB : i1 ; i2 .` **pertence à linguage**m, construindo a respectiva Árvore de Derivação.

b) Mostre que esta *GIC* **é LL(1)**.

c) Após estender a *GIC* dada, construa o respetivo **autómato LR(0)** e identifique todas as **situações de conflito** que eventualmente ocorram.

d) Usando notação do Yacc (e todas as facilidades oferecidas pelo par de ferramenta Lex/Yacc) transforme a *GIC* dada numa **gramática tradutora (*GT*)** (juntando-lhe ações semânticas) para:

d1) contar o número de instruções de cada tipo.

d2) sinalizar erro se for usado nas instruções uma variável não-declarada.

d3) sinalizar erro se for declarada uma variável repetida.