Processamento de Linguagens – MiEI

Exame de Recurso (b)

04 de Julho de 2017 (9h00)

Dispõe de **2:00 horas** para realizar este teste.

Questão 1: Expressões Regulares e Autómatos (4v)

Responda às seguintes alíneas:

a) Considere as seguintes linguagens L1, L2 e L3:
L1 é definida pela gramática:

```
S → a S b
  | acb
```

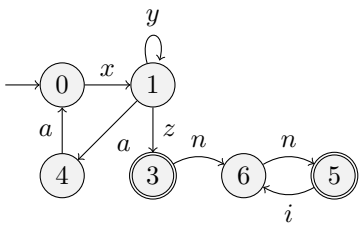
L2 é definida pela gramática:

```
S → a S
  | S b
  | acb
```

L3 é definida pela expressão regular a^+cb^+.

Compare-as entre si. Para cada par indique se são equivalentes, se uma é um subconjunto da outra, ou se são simplesmente não equivalente. (Em caso de diferenças, apresente uma frase que pertença apenas a uma delas).

b) Qual a expressão regular correspondente ao seguinte autómato:



c) O comando `sed`, disponível em Linux, permite (entre outras coisas) fazer substituições de uma expressão regular, `expReg`, por uma `string` em todas as ocorrências desse padrão num dado ficheiro de texto `file`, sendo usado do seguinte modo

```
sed -re 's/expReg/string/g' file
```

Construa, 3 comandos `sed` que, usando uma única substituição:

- Esconda todos os links dentro das âncoras HTML (`href="..."`) existentes em `file` substituindo por `href="REMOVIDO"`.
- remova os comentários inline do `C++` (`//` até ao fim de linha), incluídos em `file`.
- normalize os espaços junto às vírgulas no texto contido em `file` (p.ex. `'a  ,b'` deve ficar `'a, b'`).

d) Desenhe um autómato determinístico correspondente a: $x(bc|aa)^+b$

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

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

a) Considere o seguinte ficheiro que descreve as deslocações externas dos docentes (esquema=nome:cidade:data:descrição).

```
p1:Braga:2017-07-15:workshop A
p2:Lisboa:2017-07-15:Juri B
p3:Lisboa:2017-07-16:Ciência viva
p2:Aveiro:2017-07-17:projecto C
p2:Faro:2017-07-17:palestra D
```

Considere ainda que este ficheiro possa ter milhares de linhas.

Escreve scripts Awk para:

1. Determinar quantas vezes `p2` foi a dada cidade `C`.
2. Criar as agendas para cada pessoa: um ficheiro com o nome da pessoa (p.ex. '`p2.txt`') contendo apenas a data do evento e a cidade.
3. Verificar sobreposições garantido que *cada pessoa só pode ter uma deslocação por dia.*

b) Um sistema de suporte à decisão colaborativo foi usado entre um conjunto de 10 parceiros (Pedro, José, João, Carlos, Manuel, Mateus, Rui, Nuno, Paulo e Hugo) para votar no novo dirigente da Sociedade da qual todos são membros, sendo candidatos o Toni, o Necas, e o Xico. O sistema é muito simples: cada um dos decisores vai registando a sua opinião (positiva) a favor ou (negativa) contra um candidato. Em cada fala o decisor só exprime uma opinião sobre um candidato. As opiniões positivas são manifestadas por verbos como 'voto', 'suporto', 'a favor de', 'elejo', ou 'apoio'. As opiniões negativas são reconhecidas através de verbos como 'sou contra', 'rejeito', 'elimino', ou 'retiro'. O sistema de suporte à decisão, no fim da sessão de trabalho gera um ficheiro de texto que pode depois ser explorado com um formato como se pode ver a seguir.

```
Pedro: apoio sem duvida o Toni.
Rui: eu voto no Necas.
Carlos: nem pensar; eu rejeito de todo o Necas.
Carlos: acho que também suporto o Toni.
José: por mim sou contra o Necas.
José: e também elimino o Toni.
```

Escreva então um filtro usando o Flex para ler um ficheiro de uma sessão de votação como o listado acima e:

1. identificar todos os apoiantes do 'Toni';
2. identificar os candidatos suportados pelo 'Pedro';
3. identificar quem ganhou, ou seja, aquele que teve mais expressões positivas.

Questão 3: Desenho/especificação de uma Linguagem (3v=2+1)

Um Genealogista criou uma notação para descrever as famílias (data de casamento, pais e filhos) que vai reconstruindo, de modo a povoar depois uma base de dados. As anotações sobre as famílias seguem o estilo retratado a seguir.

```
F1/1920-03-04: (João Pedro; Dias da Silva; 1900-01-02;) +
               (Maria Ana; Lacerda; 1905-03-04; 1970-04-03) =
               <João Maria/1922-10-14, Maria/1924-12-04,
                Joana/1926-05-15, Manuel Antonio/1928-09-24>
F2/1940-08-14: (Carlos Guilherme; Sousa; 1920-01-02;1960-04-03) +
               (Sonia; Maia Lacerda; 1918-10-04; 1971-12-03) =
               <Teófilo/1942-10-14, Teotónio/1941-12-31, Tomás/1948-14-24>
```

Baseando-se nesta descrição responda às seguintes alíneas:

a) Escreva uma gramática independente de contexto (GIC) para a linguagem apresentada;

b) Especifique o respetivo analisador léxico, usando a notação do *Flex*;

Questão 4: Gramáticas, e Parsing Top-Down (4v=1+1+1+1)

Considere a gramática independente de contexto, G, abaixo apresentada, atendendo a que os símbolos terminais T e não-terminais NT são definidos antes do conjunto de produções P, sendo F o seu axioma ou símbolo inicial.

```
T  = { '!', k, m }
NT = { F, L }

p0: F  -> L '!'
p1: L  -> k L k L
p2:     | m L m
p3:     | &
```

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

a) Gere uma frase válida da linguagem definida por G de comprimento maior ou igual a 5, construindo a respectiva Árvore de Derivação.

b) Construa a Tabela de Parsing LL(1) e assinale os conflitos encontrados.
Apresente os respetivos *first()*, *follow()* e *lookahead()*.

c) Se num dado momento do `Parsing Top-Down LL(1)` a *stack de parsing* contiver os símbolos a seguir listado (topo à esquerda e `'$'`, que representa o fim de ficheiro, no fundo à direita)

```
L | k | L | m | $ |
```

diga o que significa esse estado, isto é, o que é que já foi reconhecido e qual pode ser o próximo símbolo do ficheiro de entrada para o parsing continuar sem erros.

d) Diga justificando se a função abaixo, que é pressuposto reconhecer o símbolo inicial F, pertence ao parser RD (recursivo-descendente) correto para reconhecer a linguagem gerada pela gramática G acima.

```
void recF(int Lido){
    switch (Lido) {
       case k   : recL(Lido); recT(k,Lido); recL(Lido);
                  break;
       case m   : recL(Lido); recT(m,Lido);
                  break;
       case '!' : erro(); }
}
```

Questão 5: Gramáticas, Tradução e Parsing Bottom-Up (4v=1+1+2)

A gramática independente de contexto, *GIC*, abaixo escrita em *BNF*, define uma linguagem de domínio específico para descrição de uma coleção de tuplos de facetas.
O Símbolo Inicial é `Colecao`, os Símbolos Terminais são escritos só em minúsculas (terminais-variáveis) ou entre apostrofes (sinais-de-pontuação), e a string nula é denotada por `&`; os restantes (sempre começados por maiúsculas) serão os Símbolos Não-Terminais.

```
p0:  Colecao     -->  Tuple
p1:                |  Colecao Tuple
p2:  Tuple       -->  TId '(' Faces ')'
p3:  Faces       -->  Faceta
p4:                |  Faces ',' Faceta
p5:  Faceta      -->  Field '=' Value
p6:  Field       -->  pal
p7:  Value       -->  string
p8:  TId         -->  pal
```

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

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

b) Se num dado momento do parsing Bottom-UP (BU) estiver num estado `q` e, ao ler o símbolo terminal da entrada `')'`, a tabela de decisão indicar que deve fazer uma redução pela produção 5 (`ACTION[q,')'] = red#5`), diga o que significa atingir esse estado e quantos estados vai recuar no autómato de reconhecimento (quantos símbolos tira da stack de parsing) e com que símbolo transita a seguir a reduzir.

c) 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:

c1) calcular e imprimir o tamanho da coleção (número de tuplos reconhecidos).

c2) calcular e imprimir o tamanho do maior tuplo (aquele que tem mais facetas).

c3) construir um índice invertido onde para cada valor se associem todos os identificadores de tuplos onde esse valor aparece

Questão 6: Compilação (1v)

Supondo que no seu programa em LPIS surge a seguinte atribuição

```
b = a * 3 / c;
```

e assumindo que os endereços de `a, b, c` são respetivamente `0, 1, 2`
diga justificando qual dos fragmentos de código Assembly da VM (abaixo) traduz corretamente a atribuição acima.

```
   (a)          (b)          (c)
PUSHG 0      PSHA  0      PUSHA 0
PUSHI 3      PUSHG 1      LOADN
MUL          PUSHI 3      PUSHI 3
PUSHG 2      PUSHG 2      MUL
DIV          DIV          PUSHA 2
STOREG 1     MUL          LOADN
             STOREN       DIV
                          STOREG 1
```