Processamento de Linguagens – MiEI

Exame de Recurso (a)

27 de Junho de 2017 (9h00)

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

Questão 1: Expressões Regulares e Autómatos (4v)

Responda às seguintes alíneas:

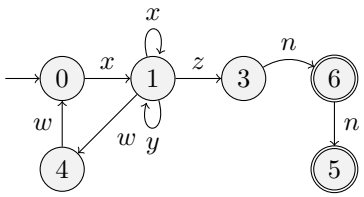
a) Considere as seguintes linguagens L1, L2 e L3:
L1 é definida pela gramática:

```
S → a S b
  | acb
```

L2 é definida pela expressão regular a^+cb^+. L3 é definida pela expressão regular $(ab)^+c$.

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

- remova os comentários `C++` (`//` até ao fim de linha), incluídos em `file`
- junte uma vírgula no final de cada linha (se ela não existir).

d) Desenhe um autómato determinístico correspondente a: $x(acd)^+j$

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) Observe o exemplo abaixo em que se mostra uma notação muito leve para descrever os triplos de uma ontologia. O 1º termo do triplo é o conceito origem da ligação, o sujeito; o 2º termo do triplo é a relação, ou predicado, (se o separador for o '-') ou é uma propriedade (se o separador for o ':'); o 3º termo do triplo, que na verdade pode ser uma lista de termos separados por ',', é o conceito destino da ligação, o objeto, ou o tipo da propriedade.

```
( casa -
    tem ->
         telhado,porta,janela  )
( casa - pertence -> dono )
( dono - é -> pessoa )
( pessoa - exerce -> profissao )
( pessoa : nome -> string )
( casa: endereço -> string )
( casa: retratada -> foto)
( pessoa:
    retratada -> foto)
```

Escreva uma script gawk que:

1. conte o número de conceitos e relações distintas encontradas (no exemplo acima, 7 conceitos e 4 relações).
2. liste cada sujeito com todas as suas propriedade (no exemplo acima o sujeito `casa` tem as propriedades `endereço` e `retratada`).
3. reescreva a entrada num formato XML como se ilustra abaixo, tendo o cuidado de separar as lista de objetos:

```
<triple>casa; tem; telhado</triple>
<triple>casa; tem; porta</triple>
<triple>casa; tem; janela</triple>
..............
<triple>pessoa; nome; string<triple>
..............
```

b) Escreva um filtro usando o Flex para ler um programa em C formado por 1 ou mais funções e para *instrumentar* cada função.
Para tanto deve acrescentar no início de cada função uma instrução de escrita do género

```
fprintf(trace,"Vai iniciar-se a execução da função %s \n",nome);
```

e no fim de cada função, antes do `return` ou da chaveta final sem `return` (quando o tipo da função é `void`), uma instrução de escrita do género

```
fprintf(trace,"Vai terminar a execução da função %s (de tipo void) \n",nome);
```

em que `trace` é uma apontador para ficheiro aberto para escrita e `nome` é uma variável que guarda o identificador da função em análise. Note ainda que o comentário '`(de tipo void)`' só pode aparecer nos casos em que o cabeçalho declare esse tipo de retorno.

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

BibTeX é uma linguagem de Domínio Específico para descrever diferentes tipos (atualmente 20 variantes) de referências bibliográficas que podem ser citadas em documentos LaTeX, conforme se mostram 3 exemplos a seguir.

```
@incollection{MHL2015a,
  author = {Ricardo Martini and Pedro Rangel Henriques and Giovani Libreloto},
  title={Storing Archival Emigration Documents to Create Virtual Exhibition Rooms},
  booktitle={New Contributions in Information Systems and Technologies},
  series={Advances in Intelligent Systems and Computing},
  editor={Rocha, Alvaro and Correia, Ana Maria and Costanzo, Sandra and Reis, Luis Paulo},
  volume={353},
  pages={403-409},
  year = {2015},
  month =   {April}
}
@InProceedings{MHC2015,
  author = {Vitor T. Martins and Pedro Rangel Henriques and Daniela da Cruz},
  title = {An AST-based tool, Spector, for Plagiarism Detection: the approach, functionality, and implementat
  booktitle = {Proceedings of the 2015 Symposium on Languages, Applications and Technologies, SLATE'15},
  pages = {173--178},
  ISBN = {},
  year = {2015},
  month =   {},
  publisher = {Fundación General UCM},
  annote = {Keywords: software, plagiarism, detection, comparison, test}
}
@book{Oli91a,
    author = "José Nuno Oliveira",
    title = "Especificação e Semântica",
    year = 1991,
    edition = "1.st",
    publisher = "Departamento de Informática, Univ. do Minho"
}
```

Baseando-se nesta descrição responda às seguintes alíneas:

a) Escreva uma gramática independente de contexto (GIC) para a linguagem apresentada considerando que o tipo de cada registo (que aparece logo no início depois de `'@'`) e o nome dos campos dos registos são variáveis;

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, que permite declarar uma ou mais variáveis definindo o seu tipo e permite executar instruções 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, cmdA, cmdB }
NT = { S, Ds, Is, As, I, IIs, Tip, Var }
P = {
p1: S  -> Ds '{' Is '}'
p2: Ds -> &
p3:     | Tip Var ';' As
p4: As -> Tip Var ';' As
p5:     | &
p6: Is -> I ';' IIs
p7: IIs-> I ';' IIs
p8:     | &
p9: I  -> cmdA Var
p10:    | cmdB Var Var
p11: Tip -> id
p12: Var -> id
}
```

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

a) Mostre que a frase `{ cmdA x ; cmdB x y; }` **pertence à linguagem**, construindo a respectiva Árvore de Derivação.

b) Construa a Tabela de Parsing LL(1) e prove que a gramática é LL(1).
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 (topo à esquerda e `'$'` a representar o fim de ficheiro)

```
Is | '}'  | $ |
```

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) Escreva as funções de um parser RD-puro (recursivo-descendente) para reconhecer os 2 símbolos não-terminais `I` e `As`.

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 listas.
O Símbolo Inicial é `Listas`, 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:  Listas      -->  Lst
p1:                |  Listas Lst
p2:  Lst         -->  '(' Args ')'
p3:  Args        -->  Arg
p4:                |  Args ',' Arg
p5:  Arg         -->  num
p6:                |  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 `ACTION` indicar que deve fazer uma redução pela produção 2 (a ação determinada for `red#2`), diga quantos estados vai recuar no autómato de reconhecimentos (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 número total de listas e o número de argumentos de cada lista.

c2) garantir que se verificam as seguintes regras semânticas:

- se o primeiro argumento de uma lista for um `id` a seguir devem aparecer 3 números;
- se o primeiro argumento de uma lista for um `num` a seguir devem aparecer tantos argumentos quantos o valor desse número.

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) seria gerado

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