Processamento de Linguagens – MiEI

Teste

29 de Maio de 2019 (13h00)

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

Questão 1: Expressões Regulares e Autómatos (3v)

Responda às seguintes alíneas:

a) Considere a seguinte expressão regular:

$$b\ (\ d\ +\ c\ )^*\ f\ g\ (\ n^2\ )^*$$

e gere os correspondentes autómatos, o não-determinista (aplicando as regras formais de transformação), e o determinista (transformando informalmente).

b) Escreva em notação do Flex mais corrente as seguintes expressões regulares:

$$a\ +\ d\ +\ f$$

$$(a\ +\ b\ +\ c\ +\ d)(a\ +\ b\ +\ c\ +\ d)(a\ +\ b\ +\ c\ +\ d)$$

$$(a\ +\ A)\ (T\ +\ t)\ (E\ +\ e)$$

Apresente todas as frases que cada uma delas *apanha*. No caso de serem muitas, apresente 3 ou 4 das mais curtas. Comente quando necessário

c) O comando `grep -Po patt file.c` permite encontrar as substrings de file.c que fazem match com `patt`. Escreva um padrão capaz de extrair todas as strings de um programa C tendo o cuidado de contemplar:
(1) linhas C contendo mais que uma string (`strcmp("abc","efg")`);
(2) strings contendo aspas protegidas (`"uma string com \" é válida"`)

Questão 2: Filtros de Texto em Flex e Gawk (6v = 3+3)

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

a) Considere o seguinte extrato de uma linguagem pypod[1]:

```
=head1 Name
 p - Este programa faz a ...

=cut

def f1(x):
   return x if x < 3 else f1(x-1)*x

def f2(y,z): return y*10+x

=head1 Synopsys
 opções
 ...
 Exemplo de como criar uma função

def f3(f): ...
=cut
```

A linguagem pypod tem as seguintes convenções:

- As funções são definidas com a keyword "def" (em início de linha).

[1] Python + Perl documentation

- Aceita comentários multilinha contidos entre "=head".... "=cut"(ambos em início de linha) que constituem o manual.

(a1) Faça um filtro Flex pypodman que extraia o manual (comentários "=head ....=cut"). Aplicado ao exemplo anterior, a saída esperada é:

```
=head1 Name
 p - Este programa faz a ...

=head1 Synopsys
 opções
 ...
 Exemplo de como criar uma função

def f3(f): ...
```

(a2) Faça um filtro Flex que envie para a stdout o índice das funções, (nome, parâmetros e número da linha associados). As funções contidas nos comentários não devem aparecer.

```
6:f1(x)
9:f2(y,z)
```

b) Para planear custos de uma urbanização, criou-se um ficheiros com o seguinte aspeto:

```
## prédios de Gualtar
vidro 20x50 :: 300 :: euros
vidro 100x80 :: 500 :: euros
fechadura :: 1000 :: euros
tijolo :: 20 :: euros
telha :: 20 :: euros

Porta     :: 4   :: vidro 20x50
Porta     :: 1   :: fechadura
Janela    :: 2   :: vidro 100x80
postigo   :: 1   :: vidro 20x50

rés do chão :: 4  :: Porta
rés do chão :: 12 :: postigo
rés do chão :: 900  :: tijolo

andar :: 4   :: postigo
andar :: 14  :: Janela
andar :: 1000  :: tijolo

telhado :: 200 :: telha
telhado :: 2 :: postigo

predio :: 1 :: rés do chão
predio :: 4 :: andar
predio :: 1 :: telhado

bairro :: 20 :: predio

Porta
bairro
```

Construa uma script em Gawk que calcule o preço dos vários elementos, e que os imprima quando requerido.

Quando uma linha têm um só campo deve escrever o preço desse elemento.

As linhas começadas por "#"são comentários e devem ser saltadas.

Saída esperada:

```
Porta ==> 2200
bairro ==> 3516000
```

Questão 3: Desenho/especificação de uma Linguagem (5v=3+1+1)

Para ensinar crianças a programar construiu-se, com base num microprocessador e alguns sensores baratos, um brinquedo móvel, um tanque de guerra, comandável por software. O dito tanque é capaz de andar até estar próximo de um obstáculo, parar definitivamente todo o programa, avançar ou recuar um determinado número de passos (o tamanho em centímetros correspondente a 1 passo também é programável), virar à esquerda ou direita um determinado número de graus, reagir com uma das ações de avanço ou viragem anteriores se ouvir um som estranho ou se ficar escuro, disparar 1 vez a arma 1 ou disparar N vezes a arma 2.

Neste contexto, responda às alíneas seguintes:

a) Escreva uma GIC (Gramática Independente do Contexto) que defina uma linguagem simples para programar o dito tanque de guerra. Essa linguagem, além de ativar as ações descritas, deve ainda permitir repetir ações (uma ou mais).

Assegure-se que a sua GIC verifica a Condição LL(1).

b) Usando a gramática especificada em cima apresente um exemplo de um pequeno programa para o dito brinquedo e construa a respetiva Árvore de Derivação.

c) Usando a notação do Flex especifique um analisador léxico para a linguagem em causa.

Questão 4: Gramáticas, Parsing e Tradução (6v=2+1+1+1+1)

Considere a Gramática Independente De Contexto, G, abaixo apresentada, que define uma linguagem para descrever (de modo muito simples) um *Blog* (sequência de *posts*).

Note ainda que os símbolos terminais T e não-terminais NT estão definidos antes do conjunto de produções P, sendo `Blog` o seu axioma (ou símbolo inicial).

```
T  = { '{', '}', '.', '(', ')', ':',  AUT, num, txt, id }
NT = { Blog, Posts, Post, Author, Likes, Coments }
P = {
p1: Blog    -> Posts
p2: Posts   -> Posts Post
p3:          | Post
p4: Post    -> Author txt Likes Coments '.'
p5: Author  -> AUT ':' id
p6: Likes   -> '(' num ')
p7: Coments -> '{' Posts '}'
p8:          | &
}
```

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

a) Calcule o *Lookahead(1)* de cada produção e diga justificando se há **Conflitos LL(1)**.

b) Após estender a G dada, construa completamente o *estado inicial* do respetivo **Autómato LR(0)** e os *estados que dele imediatamente derivam*.

c) Diga porque é que em um desses estados, seguintes ao estado inicial, ocorre uma situação de **Conflito shift-reduce** e sugira como se poderá resolver.

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

e) Transforme a GIC dada numa GT (Gramática Tradutora) desenvolvendo as ações semânticas necessárias para escrever o *identificador do autor* de cada *post* anotado com uma marca XML `'autor'` seguido do respetivo `texto` anotado com a marca XML `'txt'`, devendo cada linha dessas começar pelo número do *post*. O número do *post* é o seu número de ordem dentro de cada nível; o nível aumenta sempre que um post é um comentário a outro post.
Por exemplo, uma linha do texto de saída produzido poderia ser:

```
2.1.3 <aut>ZebraFive</aut> <txt>Eheheh, nem eu diria tão bem</txt>
```

significando que o autor `ZebraFive` comentou `Eheheh, nem...` sendo este o 3º comentário ao autor do 1º comentário ao autor do 2º *post*.