Processamento de Linguagens (LEI)

Teste (época normal)

23 de Janeiro de 2023 (15h30)

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

Questão 1: Expressões Regulares (4.5v = 1.5+1.5+1.5)

Responda a cada uma das alíneas seguintes:

a) Considere as expressões regulares seguintes (ER):

```
e1 = a (a b)+ (c d | c f)* j
e2 = (a a b)+ c (d* | f*) j
```

e mostre que e1 e e2 não são equivalente, apresentando 2 frases que derivam de ambas e depois uma frase derivada de e1 e uma frase derivada de e2 que não é válida na outra ER.

b) Especifica uma expressão regular que faça match com todas as strings binárias, compostas apenas por zeros e uns, que contenham pelo menos dois uns consecutivos;

c) Do que é que as pessoas gostam?

Escrever uma função python que dado um texto devolva a lista das palavras que se seguem a `gostar de`, (mais precisamente "gost... de/do/da/dos/das X").

Exemplo:

```
In: O Manel gosta de passear e sempre gostou da praia.
Out: [passear, praia]
```

Questão 2: módulo re (4v = 1+1+1+1)

Considere o seguinte excerto de código :

```
import re
html = '<a href="http://www.pl-uminho.pt"><h1><b>Teste de PL</b></h1></a>'
html2 = '<a href="http://www.pl-uminho.pt">Teste de PL</a>'
match = re.search(r'<a\s+href="([^"]*)">(.*?)</a>', html)
match2 = re.search(r'<a\s+href="([^"]*)">(.*?)</a>', html2)
if match:
    url = match.group(1)
    content = match.group(2)
    print(url, content)
if match2:
    url = match2.group(1)
    content = match2.group(2)
    print(url, content)
```

Para cada uma das afirmações seguintes indique se é verdadeira é falsa, no caso de ser verdadeira justifique, no caso de ser falsa apresente a correção da afirmação:

a) As instruções apresentadas extraem apenas o valor do href da tag `<a>` num primeiro grupo de captura, ficando o segundo grupo de captura vazio;

b) As instruções apresentadas extraem o valor do href da tag `<a>` num primeiro grupo de captura, ficando o segundo grupo de captura com o resto do conteúdo da tag `<a>`;

c) As instruções apresentadas imprimem para o stdout:

```
href="http://www.pl-uminho.pt"><h1><b>Teste de PL</b></h1>
```

d) As instruções apresentadas imprimem para o stdout:

```
http://www.pl-uminho.pt <h1><b>Teste de PL</b></h1>
http://www.pl-uminho.pt Teste de PL
```

Questão 3: Gramáticas (5v = 3+2)

Uma base de regras (BR) em Prolog é formada por um conjunto de regras de inferência que são implicações lógicas.

À esquerda (cabeça da regra) do operador *"é-implicado-por"* (denotado por `':-'`) surge o átomo dito consequente e à direita (corpo da regra) temos o antecedente que é uma lista de átomos logicos separados pelo operador lógico de conjunção (denotado por uma vírgula).

Cada regra termina num ponto final, como se exemplifica a seguir:

```
pred1 :- at2, at3.
pred(o1,o2) :- pred2(o1), pred3(o2), pred4(o1,o2).
pred2(X) :- pred5(X).
```

Como exemplificado, um átomo tem sempre o nome do predicado (função booleana) seguido por 0 ou mais argumentos. Os argumentos são uma lista de valores entre parêntesis e separados pela clássica vírgula.

Em Prolog um valor pode ser uma constante numérica ou alfanumérica, uma variável, uma lista (constantes ou variáveis entre parentesis retos), ou mesmo um átomo.

Neste exercício pede-se que:

a) Escreva, em BNF-puro, uma Gramática Independente de Contexto (GIC) que não tenha conflitos LL(1) para definir uma BR em Prolog;

b) Sabendo que os números podem ser inteiros ou reais e podem ser precedidos por um sinal, que as constantes alfanuméricas podem ser uma palavra (letras eventualmente com dígitos pelo meio desde que não na primeira posição) em minúsculas ou uma string entre apóstrofes, e as variáveis são palavras em maiúsculas, escreva o analisador léxico da linguagem definida na alínea anterior, associando a todos os símbolos terminais as respetivas Expressões Regulares.

Questão 4: Compilador (6.5v = 1.5+1+1.5+1+1.5)

Considere os Terminais `"str"` (texto entre apostrofes), `"real"` (número decimal) e `"id"` (sequência não nula de letras e hifens) e a seguinte Gramática Independente de Contexto (G) em que Stand é o `Axioma` e `"&"` representa a string nula.

```
T  = { '.', ';', AUTOS, BMW, VW, KIA  id, str, real }
NT = { Stand, As, Mais, Auto, Marca, Model, Preco }
P = {    p1: Stand  : AUTOS As '.'
         p2: As     : Auto Mais
         p3: Mais   : &
         p4:        | ';' Auto Mais
         p5: Auto   : Marca Model Preco
         p6: Marca  : BMW
         p7:        | VW
         p8:        | KIA
         p9: Model  : str
         p10: Preco : real          }
```

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

a) Escreva uma *frase válida* da linguagem gerada por G, apresentando a respetiva *árvore de derivação.*

b) Diga justificando se há **Conflitos LL(1)**.

c) Após estender a G dada, construa completamente o respetivo *Autómato LR(0).*
Diga, justificando, se em algum desses estados ocorrem de **Conflitos shift-reduce**.

d) Escreva 2 funções de um parser RD (recursivo-descendente), uma para reconhecer qualquer símbolo terminal e outra para reconhecer o símbolo não-terminal `Mais`.

e) Acrescente Ações Semânticas às produções da gramática para associar ao valor semântico do Axioma (em Python seria `p[0]` da produção $p1$) o número total de veículos em stand e o valor total do seu custo.