Processamento de Linguagens (LEI)

Exame de Recurso

13 de Junho de 2023 (14h30)

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

Questão 1: Expressões Regulares (3v = 1+1+1)

Responda a cada uma das alíneas seguintes:

a) Considere as expressões regulares seguintes (ER):

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

e mostre que e1 e e2 são equivalente.

b) Especifica uma expressão regular que faça *match* com todas as strings binárias, compostas apenas por zeros e uns. que se começar por '0' termine em '0' e que se começar por '1' tem de ter pelo menos mais um '1' logo a seguir.

c) Escreva uma expressão regular que aceite emails com a estrutura dos exemplos seguintes:

```
ana@gmail.com
_ana@braga.empresa.pt
ana1.maria2.silva3@hotmail.com
fred_smith2000@inst.ac.uk
```

Questão 2: módulo re (6v = 1+1+1+1+1+1)

Leia com atenção e responda aos desafios seguintes:

a) Considere o seguinte extrato de um filtro de texto em Python

```
import re
linha = input()
y = re.findall(r'[ ]+[^0-9]+\.', linha)
if(len(y)>0):
    print("existem ", len(y), " ocorrências, a primeira ", y[0])
else:
    pass
```

e responda às alíneas seguintes:

1) Diga qual a resposta dada pelo filtro acima se o texto de entrada for

```
linha = "3 hh-+. 345.ola.12. 34 rrr.1.rnhh .89.  ghhh ."
```

2) Diga qual a resposta dada pelo filtro acima se o texto de entrada for

```
linha = "ola.12. 34rrr.1.rnhh .89.ghhh.3hh-+.345."
```

b) Considere o seguinte extrato de um filtro de texto em Python

```
import re
import sys
for linha in sys.stdin:
  if ( s := re.search(r'\[([aeiou]+|[13579]*)\]', linha) ):
     print(s.group())
  else:
     print("falhou!")
```

e responda às alíneas seguintes:

1) Diga qual a resposta dada pelo filtro acima se `linha` for

```
linha = "LINHA [aa123] COM 1 marca de SUCESSO [a]ou [(222)]"
```

2) Diga qual a resposta dada pelo filtro acima se `linha` for

```
linha = "LINHA COM 1 marca de SUCESSO (a) ou (2)"
```

c) Considere o seguinte excerto de um analisador léxico:

```
import ply.lex as lex
tokens = ( 'NOME', 'ARG', 'TXT', 'PONTO' )
t_PONTO = r'.'
t_NOME  = r'\w+'
t_ARG   = r'\$\d+'
t_TXT   = r'\"[^\"]*\"'
t_ignore = ' \n\t'
def t_error(t): .....
```

e responda então às alíneas seguintes:

1) Diga justificando se concorda com a afirmação: Se no texto-fonte

```
print . $1 . match "hello"
```

se substituir o carater '.' por ',' ou por '|', o resultado do Analisador será diferente visto que o token PONTO só reconhece o carater '.'.

2) Diga justificando se concorda com a afirmação: Dado o texto-fonte

```
Bem vindo de volta, $UTILIZADOR!
```

se a palavra 'UTILIZADOR' fosse substituída por 23 o analisador reconheceria ao todo 7 símbolos terminais.

Questão 3: Gramáticas (5v = 3+2)

Desta vez a linguagem de domínio específico, DSL, que tem de desenvolver destina-se a apoiar o senhor B, um colecionador de livros (1ª edições), a organizar a sua biblioteca, descrevendo e catalogando os seus livros. Para tal, o dito senhor B, quer identificar cada livro por um código alfanumérico escolhido por si (muitas obras são antigas, anteriores ao ISBN), e precisa de indicar o título da obra, o autor, o ano de edição, além de alguns outros atributos que deixamos à sua imaginação. Os livros estão agrupados por temas (romance, poesia, teatro, filosofia, etc.) e a bibiloteca inclui um ou mais temas.

Neste exercício pede-se que:

a) Escreva, em BNF-puro, uma Gramática Independente de Contexto (GIC) que não tenha conflitos LL(1), para criar a DSL pretendida pelo senhor B, considerando que o símbolo Não-Terminal `Biblio` é o axioma (*start-symbol*) da GIC;

b) Relativamente à GIC da alínea anterior, identifique todos os símbolos Terminais, indicando para cada um se é um sinal (literal), uma palavra-reservada, ou uma classe variável (nesse caso descreva o conjunto de símbolos que inclui). Feito isso, escreva em PLY-LEX um Analisador Léxico para a DSL anterior.

Questão 4: Compilador (6v = 1.5+1+1+1+1.5)

Considere que & representa a produção vazia, que as palavras reservadas estão escrita em maiúsculas, os simais estão entre apóstrofes, as classes terminais variáveis estão escritas em minúsculas, e que os símbolos Não-Terminais são os que aparecem no lado esquerdo das produções.

Analise então as produções da Gramática Independente de Contexto, G, abaixo, que define parcialmente a linguagem Assembly de uma Máquina Virtual com 6 registos (identificados por um cardinal # seguido de um digito de 1 a 6). As labels, que opcionalmente identificam as instruções, são alfanuméricas mas têm de começar pela letra *L*. Os Endereços de memória (variáveis do programa) são números inteiros sempre relativos ao global-pointer (começam por *G*) ou ao frame-pointer (começam por *F*). As constantes podem ser números (inteiros ou reais) ou strings (entre apóstrofes).

```
p1 Ass    : Instrs
p2 Instrs : Instrs ';' Instr
p3        | Instr
p4 Instr  : Label COper Opers
p5 Label  : idinst ':'
p6        : &
p7.. COper: START | STOP | RET | ADD | MUL | INP | OUT
p14..     | JUMP | JZER | CALL | DBYTE
p18 Opers : Opers Oper
p19       | &
p20 Oper  : Const
p21       | idreg
p22       | endmem
p23       | idinst
p24 Const : num
p25       | string
```

Neste contexto, responda às alíneas seguintes:

a) Mostre que o texto-fonte seguinte

```
START Lini; DBYTE G0 10; Lini: ADD G0 43 #1; OUT #1; STOP
```

é uma *frase válida* da linguagem gerada por G, construindo a respetiva *árvore de derivação*.

b) Identifique as situações em que há **Conflitos LL(1)** e sugira, para um dos casos, uma forma de o resolver.

c) Após estender a *G* dada, construa apenas o estado inicial do respetivo *Autómato LR(0)* e os estados a ele adjacentes.

d) Supondo que era possível desenvolver um parser RD (recursivo-descendente) para a linguagem gerada por G, escreva uma função para reconhecer qualquer símbolo terminal e outra função para reconhecer o símbolo não-terminal `Label`.

e) Acrescente Ações Semânticas às produções da gramática para criar uma lista com todas as labels definidas e assinalar um erro se um operando for uma label (identificador de instrução) que nunca seja definida.