Processamento de Linguagens e Compiladores — LCC (3ºano)

Exame de Recurso

Data: 03 de Fevereiro de 2017
Hora: 09:00

Dispõe de 2 horas para realizar este exame

Questão 1: Expressões Regulares e Autómatos (5v)

Considere as seguintes ERs:

$$e1 \;=\; a\ b \;+\; (c\ d\ e^*)$$
$$e2 \;=\; a\ (b \;+\; c\ d)\ e^*$$
$$e3 \;=\; (a\ b \;+\; c\ d)^*\ e^+$$

Responda, então, às seguintes questões:

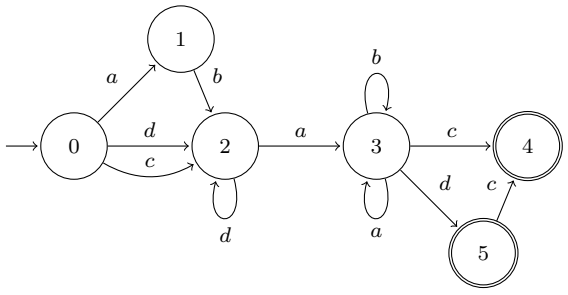
a) explique a linguagem gerada por $e1$ e, usando a respectiva *cadeia de derivação*, diga se a frase "`cdee`"pertence a essa linguagem.

b) construa informalmente o Autómato Determinista equivalente a $e2$.

c) construa formalmente (aplicando as regras dadas nas aulas) o Autómato Não-Determinista equivalente $e3$.

d) Escreva uma expressão regular para definir o *caminho para um ficheiro num sistema de diretorias*. Seguem-se alguns exemplos de caminhos válidos: "/", "/dir1", "dirA/dirB/", "/dir/dir1/dirA/fich.ext", etc;

e) Qual a expressão regular correspondente ao seguinte autómato:



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

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

a) Supondo que lhe é fornecido um ficheiro de triplos de uma ontologia muito grande (com mais de 10000 linhas) com o formato seguinte por cada linha:

```
(Sujeito,Relacao,Predicado)
```

desenvolva em GAWK um filtro que conte (e imprima no fim) as ocorrências de cada *conceito* distinto (note que os conceitos são o *Sujeito* e o *Predicado* de cada triplo) e faça uma lista de todas as *relações* mencionadas.

b) Considere a seguinte script GAWK:

```
#!/usr/bin/gawk -f
BEGIN  { RS="href=[\"']"; FS="[\"']"; }
NR > 1 { print $1}
```

Indique o que ela faz quando aplicada a um ficheiro HTML. Para ilustrar a sua resposta, escreva um pequeno exemplo HTML e a respetiva saída.

c) Especifique em FLex um normalizador de espaço branco: tabs, mudanças de linha e espaços. O texto de entrada deverá ser passado para a saída com as seguintes transformações:

- Caracteres brancos no início da linha são retirados;
- Caracteres brancos no fim da linha são reduzidos apenas à mudança de linha;
- Qualquer sequência de um ou mais caracteres brancos no meio de uma linha (tabs e espaços) é reduzida a um único espaço.
- Qualquer palavra após um sinal de pontuação deve começar com uma maiúscula.

d) Escreva um filtro FLex que, dada receba como entrada uma GIC escrita em notação Yacc, e na saída imprima uma lista com todos os símbolos terminais dessa gramática – tanto os expressamente definidos como tokens (`%token ...`), como os terminais que aparecem protegidos nas produções da gramática (exemplo ','').

Questão 3: Desenho/especificação de uma Linguagem (3v)

Pretende-se uma linguagem de Domínio Específico que permita descrever os *Achados* de uma Escavação no âmbito de um processo Arqueológico. Para tal deve-se começar por identificar a escavação (nome, local). Depois descrevem-se os arqueólogos intervenientes (código, nome, unidade a que pertence, função na equipa (diretor, investigador, ajudante), e opcionalmente a área de especialidade (vidros, moedas, edifícios, metais)). Deve a seguir ser possível descrever os objetos encontrados (os ditos *Achados*) identificados através de um código, um tipo (vidro, moeda, edifício, metal), uma descrição (texto livre), o arqueólogo que o achou, o arqueólogo que o identificou, as coordenadas geográficas da local de descoberta. Por cada objeto deve ser ainda indicado o seu estado (mau, razoável, bom).

Escreva então uma Gramática Independente de Contexto, *GIC*, que especifique a Linguagem pretendida (note que o estilo da linguagem (mais ou menos verbosa) e o seu desenho são da sua responsabilidade).

Questão 4: Gramáticas, Tradução e Parsing Bottom-Up (8v)

Considere a gramática independente de contexto, *GIC*, abaixo apresentada, que permite descrever um processo de partilha dos bens herdados. A *GIC* abaixo permite descrever inicialmente o lote de bens (objetos) a partilhar; depois descrevem-se os herdeiros envolvidos e por fim indica-se, por cada herdeiro, a lista de bens escolhidos.

Note ainda que os símbolos terminais T estão definidos antes do conjunto de produções P, e os símbolos não-terminais NT são os que estão no lado esquerdo das produções. o símbolo `Partilhas` é o axioma da gramática (ou símbolo inicial).

```
T  = { '.', ';', ':', '(', ')', '>', ',', cod, str, num, A, B, C }
P = {
p1: Partilhas  -> Bens ':' Herds ':' Escolhas
p2: Bens -> Bem OBens
p3: OBens -> ';' Bens
p4:        | &
p5: Herds -> '(' Herd ')' OHerds
p6: OHerds -> Herds
p7:        | &
p8: Escolhas -> Esc Escolhas
p9:          |  '.'
p10:Esc   -> cod '>' Lst
p11:Lst   -> cod '-' Pref RLst
p12:RLst  -> &
p13:       | ',' cod '-' Pref RLst
p14:Pref  -> A
p15:       | B
p16:       | C
p17:Bem   -> cod '-' Desc '-' Valor
p18:Valor -> num
p19:Desc  -> str
p20:Herd  -> cod ',' Nome ',' Contacto
p21:Nome  -> str
p22:Contacto -> str
}
```

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

a) Construindo a respectiva Árvore de Derivação, dê exemplo de uma frase concreta que pertença a esta linguagem.

b) Mostre que esta *GIC* **é LL(1)** provando que não há conflitos em nenhuma das alternativas.

c) Escreva as funções de um Parser RD para reconhecer qualquer *Terminal* e para Reconhecer os símbolos `Herds` e `OHerds`.

d) Após estender a *GIC* dada, construa o respetivo **autómato LR(0)** — limite o seu desenho ao estado inicial, aos estados que dele saem e aos estados a seguir a cada um desses.

e) Se num dados momento do `Parsing Bottom-Up (BU)` a *stack de parsing* contiver os símbolos (topo à esquerda)

```
Herds | ':' | Escolhas | $ |
```

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.

f) Usando notação do Flex escreva um Analisador Léxico para a *GIC* dada, passando através de `yylval` o valor dos símbolos que representam classes de terminais.

g) 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:

g1) contar o número de objetos a partilhar e o valor total desses bens e o valor que cabe a cada herdeiro (se todos receberem o mesmo).

g2) sinalizar erro se algum herdeiro escolher um objeto não-declarado.

Note que basta escrever as produções que tiver de completar com ações semânticas.