Processamento de Linguagens I
LESI + LMCC (3º ano)

5º Ficha Prática

Ano Lectivo de 04/05

1 Objectivos

Este ficha prática contém exercícios para serem resolvidos nas aulas teórico-práticas com vista a sedimentar os conhecimentos relativos a

- Análise de Gramática para construção de Tabelas de Parsing LL (*Top-Down*); verificação da Condição LL(1).
- Análise de Gramática para construção de Tabelas de Parsing LR (*Bottom-Up*); verificação da Condição LR(0)/SLR(1).

2 Enunciados

No contexto do desenvolvimento de Compiladores, ou mais genericamente de Processadores de Linguagens, a primeira tarefa a realizar é a criação de uma GIC que especifique a linguagem que se quer processar, visto que todo o desenvolvimento vai assentar sobre a dita gramática.
Em particular, o **reconhecedor sintáctico (parser)** é construído a partir de uma algoritmo iterativo genérico, muito eficiente e independente da gramática, que é guiado por uma **tabela de decisão**, a qual é específica de cada linguagem e se cria sistematicamente a partir da GIC concreta.
Nessas condições, propõe-se para esta aula a construção da tabela LL(1) e da tabela LR(0), ou SLR(1), a partir da gramática indicada em cada exercício, a verificação das condições de reconhecimento para cada uma das estratégias (TD ou BU) e a eventual transformação da GIC.

2.1 Horto de Braga

Determinado horto desta cidade faz propostas de fornecimento de plantas (árvores e arbustos) para construir, ou reconstruir, jardins exteriores, públicos ou particulares. Nesse contexto pretende-se desenvolver um processador de linguagens para implementar diversas operações associadas à gestão corrente do Horto.
Analise atentamente a seguinte gramática independente de contexto, que está simplificada por razões óbvias. O Símbolo Inicial é `Flores` e os Símbolos Terminais são escritos em minúsculas (pseudo-terminais), ou em maiúscula (palavras-reservadas), ou entre apostrofes (sinais de pontuação). A string nula é denotada por `&` e o caracter `$` representa o fim-de-ficheiro (do texto de entrada).

```
p1:  Flores    -->  FsExt  FsInt
p2:  FsExt     -->  FEXTERIOR  Fs
p3:  FsInt     -->  &
p4:               |  FINTERIOR
p5:  Fs        -->  Flor  MaisFs
p6:  MaisFs    -->  &
p7:               |  ","  Fs
p8:  Flor      -->  Cod  NomVulgar  Preco
p9:  NomVulgar-->  str
p10: Preco     -->  num
p11: Cod       -->  pal
```

Calcule então as Tabelas acima pedidas.

2.2 Documento anotado em XML

Neste exercício retoma-se o problema 2.3 da Ficha de Prática nu.º 3, em que se pedia para desenvolver um processador para Documentos Anotados.
Considere a seguinte gramática independente de contexto como uma possível solução para os requisitos impostos nesse enunciado quanto ao conceito de Documento XML.

```
p1:  DocXML    -->  MarcaAbr  Conteudo  MarcaFec
p2:  MarcaAbr  -->  "<"  EleXML  ">"
p3:  MarcaFec  -->  "<"  "/"  id  ">"
p4:  EleXML    -->  id  Atribs
p5:  Atribs    -->  &
p6:  Atribs    -->  Atribs  id  "="  str
p7:  Conteudo  -->  &
p8:  Conteudo  -->  Conteudo  Componente
p9:  Componente-->  pcdata
p10: Componente-->  DocXML
```

Admita que: o Símbolo Inicial é `DocXML`; os Símbolos Terminais são escritos em minúsculas (pseudo-terminais), ou em maiúscula (palavras-reservadas), ou entre aspas (sinais de pontuação); a string nula é denotada por `&` e o caracter `$` representa o fim-de-ficheiro (do texto de entrada).
Calcule então as Tabelas acima pedidas.

2.3 Documentos sobre o Clero Catedralício

De modo a facilitar o trabalho de uma equipa de historiadores, que está a levantar informação, nos vários arquivos do País, sobre individualidades do Clero Catedralício português, definiu-se uma linguagem que permite processar automaticamente (no sentido de validar, normalizar e criar uma base de dados) as notas extraídas de cada documento consultado.
Mostra-se abaixo a respectiva gramática independente de contexto, na qual: o Símbolo Inicial é `Documentos`; os Símbolos Terminais são escritos em minúsculas (pseudo-terminais), ou em maiúscula (palavras-reservadas), ou entre aspas (sinais de pontuação).

```
p1:  Documentos -> Doc
p2:             |  Documentos Doc
p3:  Doc        -> "{" IdentDoc  IdentClerigos  Evento "}"
p4:  IdentDoc   -> Arq  CotaDoc  DataRedaccao  DataConsulta  Redactor  Tipo
p5:  IdentClerigos -> Clerigo
p6:             |  IdentClerigos ";" Clerigo
p7:  Clerigo    -> Nome  Apelido Diocese CategEcles  Papel
```

```
p8 a
p13: Evento, Arq, Redactor, Nome, Apelido, Diocese -> str
p14 a
p16: CotaDoc, Tipo, Papel  -> pal
p16: CategEcles -> BISPO
p17:            |  CONEGO
p18:            |  PRESBITERO
p19: DataRedaccao -> data
p20: DataConsulta -> data
```

Assumindo que o caracter `$` representa o fim-de-ficheiro (do texto de entrada), calcule então as Tabelas acima pedidas.