Processamento de Linguagens

3ª Ficha para as Aulas Práticas

Objectivos

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

- uso de Gramáticas Independentes de Contexto (GIC) para definir a sintaxe de Linguagens;
- uso de Gramáticas Tradutoras (GT) para definir a semântica estática e dinâmica de Linguagens;
- uso de GT para desenvolver programas eficientes, baseados em algoritmos standard guiados por Tabelas de Decisão (construídas a partir de Autómatos Finitos Deterministas com stack), para reconhecer e processar Linguagens, desencadeando Acções Semânticas específicas ao reconhecer as produções da gramática;
- geração automática de programas a partir de especificações formais;
- uso da ferramenta Yacc, disponível em ambiente Linux, para geração automática de processadores de linguagens, nomeadamente para criação de *Tradutores Dirigidos pela Sintaxe* (*Analisadores Sintácticos* e *Analisadores Semânticos*).

1 Desenvolvimento de Tradutores Dirigidos pela Sintaxe

No contexto do desenvolvimento de Compiladores, ou mais genericamente de Processadores de Linguagens, o primeiro nível, ou tarefa a implementar, é a **análise léxica** que tem por missão ler o texto fonte e converter todas as palavras correctas em símbolos terminais dessa linguagem. O segundo nível é a **análise sintáctica** que pega nos símbolos recebidos do AL e verifica se a sequência em causa respeita as regras de derivação, ou produções, da gramática. O terceiro nível é a **análise semântica** que calcula o valor, ou significado, exacto da frase e, então, valida se a sequência de símbolos sintacticamente correcta cumpre todas as condições de contexto que a tornam semanticamente válida. O quarto nível é a **tradução** que pega no significado exacto da frase válida e constrói, ou calcula, o resultado final.
Com esse fim em vista e assumindo que o 1º nível já está implementado (graças ao uso do Flex para gerar o **Analisador Léxico (AL)**), propõe-se para esta aula o recurso à ferramenta Yacc para gerar os **Analisadores Sintáctico e Semântico** e o **Tradutor** a partir da Gramática Tradutora da Linguagem a processar.
Para cada um dos exercícios, proceda em três etapas:

1. escreva a GIC e gere o *Parser* (Analisador Sintáctico) que lhe permitirá, apenas, verificar a correcção sintáctica das frases;
2. escreva, depois, uma primeira versão da GT, acrescentando à GIC anterior as Acções Semânticas necessárias para, apenas, validar a correcção semântica das frases;
3. escreva, a seguir, uma versão final da GT, acrescentando à GT anterior as Acções Semânticas necessárias para calcular e escrever o resultado desejado.

1.1 Lista de Notas

Comece por escrever uma Gramática Independente de Contexto para uma linguagem que permita escrever uma lista de notas (números inteiros sem sinal) começada pela palavra reservada NOTAS e terminada por ".", separando os

números por ",".
Serão frases válidas dessa nova linguagem apenas as que seguirem o formato dos exemplos abaixo

```
NOTAS 1 .
NOTAS 1,2, 3, 4.
```

A partir dessa GIC pretende-se que resolva as seguintes alienas:

a) Desenvolva um parser (recorrendo ao par Lex/Yacc) para reconhecer as frases válidas dessa linguagem

b) Acrescente Acções Semânticas ao dito Parser (isto é, transforme a GIC numa Gramática Tradutora (GT)), para calcular e imprimir o comprimento da lista, ou seja o número de notas indicadas.

c) Transforme a GT anterior de modo a calcular e imprimir a soma de todas as notas (em vez do comprimento).

d) Transforme de novo a GT anterior de modo a calcular e imprimir a média final de todas as notas.

e) Transforme agora a própria GIC base de modo a passar a aceitar frases do tipo

```
NOTAS ana(1) .
NOTAS ana(1); joana(2, 3, 4) .
```

e a imprimir a média das notas por cada aluno.

f) Transforme então a GT anterior de modo a imprimir também o nome do aluno antes da respectiva média final.

g) Transforme de novo a GT anterior de modo a gerar uma página HTML com essa tabela de nomes de alunos seguidos da respectiva média final.

1.2 Linguagem de Programação

Em determinada Linguagem de Programação, chamada nLP, não existem tipos pré-definidos; o programador tem de declarar o nome de cada tipo que quiser usar, indicando o seu tamanho (número de bytes que ocupa em memória). Todas as variáveis que a seguir sejam declaradas terão de ser de um dos tipos definidos. Essas variáveis serão alocadas, pelo compilador, em memória sequencialmente, a partir do endereço 0. A gramática independente de contexto G, abaixo apresentada, define a sub-linguagem de nLP para declaração de tipos e variáveis. O Símbolo Inicial é nLPD, os Símbolos Terminais são escritos em minúsculas (pseudo-terminais), ou em maiúsculas (palavras-reservadas), ou entre apostrofes (sinais de pontuação) e a string nula é denotada por &; os restantes serão os Símbolos Não-Terminais.

```
p1: nLPD   --> Tipos Vars
p2: Tipos  --> TIPOS Ts
p3: Ts     --> Tipo
p4:          | Ts Tipo
p5: Tipo   --> IdT Tam
p6: Vars   --> &
p7:          | VARIAVEIS Vs
p8: Vs     --> Ids ':' IdT
p9:          | Vs Ids ':' IdT
p10: Ids   --> IdV RIds
p11: RIds  --> &
p12:         | ',' Ids
p13: IdT   --> id
p14: Tam   --> num
p15: IdV   --> id
```

a) Transforme G numa gramática tradutora, GT, reconhecível pelo yacc, para traduzir o bloco de declaraçõoes num conjunto de factos em Prolog que sejam instâncias dos dois seguintes predicados

```
tipo( idtipo, tam ).
variavel( idvar, idtipo, endr ).
```

Note que os endereços a atribuir a cada variável terão de ser gerados pelo seu tradutor de acordo com o critério acima.

b) Estenda a GT anterior para calcular o espaço de memória total ocupado por todas as variáveis declaradas num determinado texto fonte.

Além desse resultado, deve incluir condições de contexto para garantir que:

(a) não haja re-declaraçãao de identificadores (note que os identificadores das variáveis também não podem ser iguais aos identificadores dos tipos);

(b) o tipo de todas as variáveis declaradas exista (isto é, tenha sido previamente declarado).

1.3 Documento anotado em XML

Neste caso pretende-se processar Documentos XML—textos vulgares semeados de anotações, ou marcas, tal como descrito no exercício 2.4 da Ficha 2.
Tomando por base a descrição de Documento XML aí apresentada, pretende-se desenvolver um programa que valide se toda a marca que abre fecha, pela ordem correcta (uma marca aberta dentro de outra, terá de fechar antes da primeira), e que liste todas as marcas encontradas indicando a frequência de cada uma (número de ocorrências sobre o total de marcas). O processador também deve verificar que a mesma marca abre sempre associada ao mesmo conjunto de atributos. Se o documento-fonte for válido, deve então ser gerada uma versão LaTeX em que cada fragmento de texto marcado é assinalado entre chavetas precedidas por um comando LaTeX cujo nome é igual ao nome do elemento.
No contexto desta aula o que se pretende é: que desenvolva, com auxílio do Gerador Yacc, o processador (reconhecedor+calculador e verificador) pretendido, tomando por base a gramática de um Documento XML criada na aula anterior (Ficha Prática 1); utilize o AL desenvolvido também nessa aula.