

# Processamento de Linguagens I

## LESI + LMCC (3º ano)

5º Ficha Prática

Ano Lectivo de 05/06

### 1 Objectivos

Esta ficha prática contém exercícios para serem resolvidos nas aulas teórico-práticas com vista a consolidar definitivamente os conhecimentos práticos relativos a:

- uso de Gramáticas de Atributos (GA) para definir a sintaxe e semântica (estática e dinâmica) de Linguagens;
- especificação de processadores de linguagens segundo o paradigma da *Tradução Dirigida pela Semântica*.

### 2 Tradutores Dirigidos pela Semântica

Na medida em que esta ficha tem por objectivo reforçar os conhecimentos e aptidões que os alunos devem ter adquirido ao resolver a Ficha 2, começa-se por relembrar os conceitos básicos aí introduzidos.

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** (também designada por *semântica dinâmica*) que pega no significado exacto da frase válida e constrói, ou calcula, o resultado final. A **Tradução Dirigida pela Semântica** baseia-se na especificação formal da sintaxe e da semântica da linguagem através de uma **Gramática de Atributos** e consiste em implementar cada um dos Analisadores e o Tradutor em módulos separados, que funcionarão sequencialmente (em *pipe-line*), usando a **Árvore de Sintaxe abstracta Decorada** (ASaD) como representação interna do significado do texto-fonte a processar.

A *estrutura e conteúdo* dos nodos dessa ASaD é descrita pela GIC contida na GA e pelos Atributos associados a cada símbolo da GA; o seu *processamento*—**decoração** (cálculo dos valores dos atributos em cada nodo), **validação** (verificação da sua conformidade com as regras semânticas) e **transformação** (produção do resultado final esperado)—é feito de acordo com as regras de cálculo dos atributos, as condições de contexto e as regras de tradução incluídas na dita GA.

Para resolver o exercício proposto, proceda em etapas:

1. escreva a GIC; identifique os símbolos terminais e descreva-os à custa de Expressões Regulares;

2. escreva, depois, uma primeira versão da GA, acrescentando à GIC anterior os atributos que devem ser associados a cada símbolo e as regras de cálculo que definem como determinar o respectivo valor no contexto de cada produção;
3. acrescente depois as condições de contexto que devem ser verificadas em cada produção para que a frase tenha um significado correcto e preciso;
4. escreva, a seguir, uma versão final da GA, acrescentando à GA anterior as regras de tradução que, usando os valores calculados para os atributos, são necessárias para calcular e escrever o resultado desejado.

## 2.1 Gerador de Índices Remissivos

Retome o exercício da Ficha 3 em que se pedia para gerar automaticamente, a partir de uma descrição como a que se ilustra abaixo, o Índice Remissivo a inserir no fim de um determinado documento. Recordar-se que um Índice Remissivo é uma lista, ordenada alfabeticamente, de todos os termos a destacar num documento, com a indicação das páginas do documento onde cada termo foi definido.

A descrição inicial, escrita de acordo com uma linguagem específica que deverá ser definida para o efeito, permite indicar para cada página (referida pelo seu número ou nome do apêndice) a lista de termos a destacar. Abaixo ilustra-se uma frase válida da dita Linguagem:

INDICE

```

1 = processador, linguagem, compilador
2 = compilador, interpretador, gramática
3 = gramática, GR, GIC
4 = gramática, linguagem, reconhecedor
A = gramática, YACC
B = LRC

```

FIM DO INDICE

O Índice que deve ser gerado para este exemplo é apresentado a seguir:

ÍNDICE REMISSIVO

```

compilador: 1, 2.
GIC: 3.
GR: 3.
gramática: 2, 3, 4, Apx A.
interpretador: 2.
linguagem: 1, 4.
LRC: Apx B.
processador: 1.
reconhecedor: 4.
YACC: Apx A.

```

No contexto desta aula o que se pretende é: que escreva a Gramática de Atributos que define a linguagem pretendida e que sintetiza (como atributo do Axioma) o Índice Remissivo pretendido, garantindo que não há números de páginas (ou letras de apêndices) repetidos no ficheiro de entrada.

## 2.2 Lavanda

Lavanda é uma Linguagem de Domínio Específico (DSL) que se destina a descrever as remessas de sacos de roupa que os Pontos de Recolha (PR) de uma Lavandaria enviam diariamente à Central (LC) para lavar. Cada saco tem um número de identificação e o nome do cliente que o deixou; o seu conteúdo está dividido em lotes. Cada lote corresponde a um tipo de peça (roupa-de-corpo, ou

roupa-de-casa), um tipo de tinto (branco, ou cor) e um tipo de fio (algodão, lã e fibra), registrando-se então o número de peças entregues que pertencem a esse lote.

A gramática independente de contexto  $G$ , abaixo apresentada, define a linguagem **Lavanda** pretendida. O Símbolo Inicial é **Lavanda**, os Símbolos Terminais são escritos em minúsculas (pseudo-terminais), ou em maiúscula (palavras-reservadas), ou entre apostrofes (sinais de pontuação) e a string nula é denotada por  $\&$ ; os restantes serão os Símbolos Não-Terminais.

```

p1: Lavanda --> Cabec Sacos
p2: Cabec   --> data IdPR
p3: Sacos   --> Saco '.'
p4:         | Sacos Saco '.'
p5: Saco    --> num IdCli Lotes
p6: Lotes   --> Lote Outros
p7: Lote    --> Tipo Qt
p8: Tipo    --> Classe Tinto Fio
p9: Outros  --> &
p10:        | ';' Lotes
p11: IdPR   --> id
p12: IdCli  --> id
p13: Qt     --> num
p14,15: Classe--> corpo | casa
p16,17: Tinto --> br | cor
p18,19,20: Fio --> alg | la | fib

```

Depois de transformar  $G$  numa gramática independente de contexto abstracta  $Gabs$  (pode reduzir algumas produções que lhe pareçam supérfluas), escreva uma **Gramática de Atributos**,  $GA$ , para:

- calcular (e imprimir) o número total de sacos enviados e número de lotes de cada cliente.
- calcular e imprimir o total de peças de cada um dos 12 tipos de lotes (desde 'corpo/br/alg' até 'casa/cor/fib') enviadas, para lavar, pelo PR à LC.
- calcular o custo total de cada saco, supondo que inicialmente é fornecida a tabela de preços do PR em causa (a tabela indica o preço por peça para cada tipo de lote).

A  $GA$  deve detectar situações de erro em que o número de identificação do saco seja repetido e deve sinalizar sempre que apareça um saco para um cliente já encontrado.

## 2.3 A Linguagem de Programação nLP

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 maiúscula (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

```

Depois de transformar  $G$  numa gramática independente de contexto abstracta  $G_{abs}$  (pode reduzir algumas produções que lhe pareçam supérfluas), escreva uma **Gramática de Atributos**,  $GA$ , para:

- para traduzir o bloco de declarações 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.

- calcular o **espaço de memória total** ocupado por todas as variáveis declaradas num determinado texto fonte.

A sua  $GA$ , além das *regras para o cálculo dos atributos necessários* e para apresentar o resultado desejado (ditas *regras de tradução*), deve incluir *condições de contexto* para garantir que:

- não haja re-declaração de identificadores (note que os identificadores das variáveis também não podem ser iguais aos identificadores dos tipos);
- o tipo de todas as variáveis declaradas exista (isto é, tenha sido previamente declarado).