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

1º Ficha Prática

Ano Lectivo de 05/06

1 Objectivos

Este ficha prática contém exercícios para serem resolvidos nas aulas teórico-práticas (tp1 e tp2) com vista a sedimentar os conhecimentos relativos a:

- uso de Expressões Regulares para definir (gerar) Linguagens Regulares;
- uso de Expressões Regulares para desenvolver programas eficientes, baseados em algoritmos standard guiados por Autómatos Finitos Deterministas, para reconhecer Linguagens Regulares;
- uso de Autómatos Deterministas Reactivos, para processar Linguagens Regulares, isto é para desencadear Acções específicas ao reconhecer frases que derivam de Padrões (definidos com base em ERs) —princípio da Programação baseada em regras *Condição-Reacção*;
- geração automática de programas a partir de especificações formais;
- uso da ferramenta Flex, disponível em ambiente Linux, para geração automática de processadores de linguagens regulares, nomeadamente para criação de *Filtros de Texto* ou de *Analisadores Léxicos*.

2 Filtros de Texto

Para introduzir a ferramenta de geração de programas FLex baseada em especificações com Expressões Regulares, e para sedimentar os conhecimentos que adquiriu sobre autómatos deterministas reactivos como suporte à construção de programas eficientes, propõem-se alguns exercícios, para resolver dentro ou fora da aula, que visam a criação de programas autónomos para *filtrar textos* (FT).

2.1 Processador de Questionários

Suponha que ao fim de cada entrevista um Repórter produz um texto com as perguntas e respostas, distinguindo umas das outras porque as perguntas começam com "EU:", no início da linha, e as respostas começam com "ELE:", também no início da linha.
Nesse contexto, pretende-se desenvolver um FT para processar os questionários que:

a) simplesmente retire do texto original as tais marcas "EU:" e "ELE:", devolvendo todo o resto da entrevista sem qualquer alteração.
Melhore o filtro, de modo a tratar as marcas, quer estejam escritas em maiúsculas, quer em minúsculas;

b) substitua a marca "EU" pela palavra "Entrevistador" e a marca "ELE" por "Entrevistado";

c) substitua a marca "EU" pelo nome do entrevistador e a marca "ELE" pelo nome do entrevistado, supondo que no início encontrará as respectivas definições (ordem irrelevante) na forma "EU=nome." ou "ELE=nome."[1]

2.2 Expansor de Abreviaturas

Quando se retiram apontamentos, ou de uma forma geral, se tem de escrever muito depressa, é hábito usar abreviaturas que correspondam a uma ou mais palavras vulgares e longas.
Suponha que criou esse costume e resolveu inserir nos seus textos as ditas abreviaturas (2 ou mais letras) precedidas pelo carácter "\". Por exemplo: "\qq"(qualquer), ou "\mb"(muito bom), ou ainda "\sse"(se e só se).
Desenvolva, então:

a) um FT que lhe devolva o texto original mas com todas as abreviaturas (que definiu à partida) devidamente expandidas;

b) melhore o seu filtro de modo a contemplar ainda o tratamento do carácter "/" no fim de uma palavra, representando o sufixo "mente", e o carácter "~"no início de uma palavra, representando o prefixo "in". Uma palavra pode conter ambos os caracteres, um no início e outro no fim (pense na abreviatura da palavra "infelizmente");

c) outra melhoria que poderia introduzir no seu filtro era contemplar a possibilidade de definir abreviaturas dentro do próprio texto, na forma "\def:abrev=termo-expandido;". Pense como o fazer e nas implicações que tal requisito teria no seu filtro original.[2]

2.3 Normalizador de Emails

Os Emails escritos à moda PRH caracterizam-se por terem todas as palavras começadas por letras minúsculas, à excepção dos nomes próprios e siglas.
Desenvolva, então:

a) um FT que normalize o texto, *capitalizando* (escrevendo a letra inicial em maiúsculas) todas as palavras no início de cada frase. Além da primeira palavra do texto, uma frase começa depois de um '.', '?' ou '!', seguido de zero ou mais espaços, eventualmente um ou mais fim-de-linha;

b) complete a especificação anterior de modo a que o seu *normalizador de emails prh* conte também todos os nomes próprios (palavras começadas por maiúsculas) e siglas (palavras formadas só por maiúsculas, uma ou mais) encontradas no texto original.

3 Analisadores Léxicos

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 (que se pretende *transformar*, caso seja uma *frase válida* da linguagem em causa) e converter todas as palavras correctas em símbolos terminais dessa linguagem.
Com esse fim em vista, propõe-se para esta aula o recurso à ferramenta Flex para gerar um **Analisador Léxico** (AL) a partir da descrição dos símbolos terminais de uma linguagem e sua associação aos respectivos códigos internos.

[1] Alínea proposta para pensar fora da aula.
[2] Alínea proposta para pensar fora da aula.

3.1 Máquina de Venda de Chocolates

Considere uma situação em que se pretende simular o funcionamento de uma máquina de vender chocolates. Dado o stock no início do dia (nome, preço e quantidade de cada produto disponível), a quantia inicial de trocos e os registos das vendas diárias (nome do chocolate escolhido e a quantia introduzida), o objectivo é calcular *a evolução do stock ao longo do dia e o dinheiro acumulado.* A animação pretendida deverá mostrar através de desenhos o estado da máquina (stock existente e dinheiro ganho) após cada movimento.
No contexto desta aula o que se pretende é: que defina uma linguagem para descrever o estado inicial da máquina e os registos de vendas efectuadas durante o dia; e que desenvolva um AL para reconhecer todos os símbolos terminais dessa linguagem e devolver os respectivos códigos.
Melhore o seu AL suportando cada *palavra-chave*, ou *palavra-reservada*, da linguagem em *maiúsculas* ou *minúsculas* e permitindo a inserção de *linhas de comentário* no meio de uma frase válida.

3.2 Anuário dos Medicamentos brancos

Considere agora uma outra situação em que, para auxiliar o Instituto Farmacêutico do Ministério da Saúde na gestão do novo lote de medicamentos brancos, se pretende criar um sistema de consulta a esses medicamentos acessível a qualquer farmácia via um browser HTML. Esse sistema deve mostrar a informação agrupada por: classe de medicamentos no Symposium Terapêutico (uma página por classe, com os medicamentos ordenados alfabeticamente); ou por fabricante (uma página única, com os medicamentos agrupados por fabricante).
Sobre cada medicamento é fornecida a seguinte documentação: nome, código, classe, composição química, preço recomendado, fabricantes disponíveis e lista de medicamentos de marca equivalentes (respectivo nome e fabricante).
No contexto desta aula o que se pretende é: que defina uma linguagem para descrever a informação envolvida no lote de medicamentos a considerar (essa linguagem terá que permitir definir inicialmente o ano a que o Symposium Terapêutico diz respeito e a lista das classes de medicamentos); e que desenvolva um AL para reconhecer todos os símbolos terminais dessa linguagem e devolver os respectivos códigos.
Melhore o seu AL suportando cada *palavra-chave*, ou *palavra-reservada*, da linguagem em *maiúsculas* ou *minúsculas* e permitindo a inserção de *linhas de comentário* no meio de uma frase válida.

3.3 Documento anotado em XML

Como sabe um Documento XML é um texto vulgar semeado de anotações, ou marcas, que são identificadores especiais (designados por *elementos XML*) intercalados entre os caracteres "<" e ">". Num documento XML bem formado, a cada *marca de abertura* corresponderá uma *marca de fecho*, que tem o mesmo identificador, mas que começa por "</" terminando na mesma em ">".
Dentro de cada *marca de abertura*, além do identificador do elemento, ainda podem aparecer triplos formados por um outro identificador (de atributo), pelo sinal "=" e pelo respectivo valor que é qualquer texto entre aspas.
Cada fragmento do documento (texto livre) entre marcas deve ser considerado em bloco como sendo o símbolo PCDATA.
Desenvolva então um AL que receba um documento XML e devolva todos os símbolos terminais encontrados, a seguir resumidos: "<", ">", "</", "=", identificador (qualquer palavra formada por letras), valor, PCDATA.

Como complemento a este exercício[3], desenvolva um filtro de texto (FT) que receba um documento XML e:

a) devolva o texto original, após ter retirado todas as marcas;

[3] Alíneas propostas para pensar fora da aula.

b) conte o número de *marcas de abertura* e o número de *marcas de fecho*, indicando *erro* sempre que se verifique um desequilíbrio entre ambas[4];

c) verifique a concordância entre as *marcas de abertura* e as *marcas de fecho*, isto é, garanta que as marcas se fecham por ordem inversa que se abrem[5]. No fim produza uma listagem, ordenada alfabeticamente, de todos os elementos distintos encontrados.

[4]Aqui apenas se pede que detecte o erro por contagem e não atendendo ao *identificador do elemento* em causa em cada marca.

[5]Mas agora tomando em atenção o *identificador do elemento* em causa em cada marca.