

Ficha prática 2

Compiladores

2018/2019

Lex (continuação)

1 Introdução

Num analisador lexical, existem por vezes situações em que o comportamento deve depender do contexto. Por exemplo, se estivermos a interpretar código que possa estar em duas linguagens distintas (por exemplo, Java e Javadoc...), é necessário produzir os efeitos corretos (e distintos) para cada um dos casos. Para que tal aconteça, é fundamental que haja delimitadores que permitam distinguir os blocos de código. No caso do Java e Javadoc, a linguagem Java é assumida como *default* num ficheiro Java. Para inserirmos comandos Javadoc, temos que colocar sempre uma entrada `/**`, a partir da qual sabemos que, até aparecer um `*/`, estamos numa área de Javadoc (veja na Figura 1 um exemplo de Javadoc e Java).

Para lidar com situações deste género, tipicamente recorre-se a *start states*. Um *start state* corresponde a um *estado* do analisador e serve para o utilizador definir ações dependentes do contexto. Para utilizar *start states*, é necessário acrescentar uma linha na secção de definições da especificação; para definir dois estados, `estado1` e `estado2`, a linha a acrescentar seria:

```
%X ESTADO1 ESTADO2
```

A utilização de estados respeita um conjunto de normas. No início de uma regra, pode-se incluir um estado (significando que essa regra só deve ser considerada se o autómato estiver nesse estado) da seguinte forma:

```
<ESTADO1>abc      printf("estou no estado1 e encontrei uma string abc");
```

Para obrigar o algoritmo a mudar para um estado qualquer, utiliza-se a instrução `BEGIN <estado>`. Para voltar ao estado inicial (ou default do lex), faz-se `BEGIN 0`.

Usando *start states*, podemos definir com exatidão o comportamento de acordo com o contexto, por exemplo para deteção de erros (podemos por exemplo fazer deteção de erros lexicais para Javadoc, diferenciada da deteção de erros lexicais em Java).

```

/**
 * Returns an Image object that can then be painted on the screen.
 * The url argument must specify an absolute @link URL.
 * The name argument is a specifier that is relative to the url argument.
 * <p>
 * This method always returns immediately, whether or not the image exists.
 * When this applet attempts to draw the image on the screen, the data will be loaded.
 * The graphics primitives that draw the image will incrementally paint on the screen.
 *
 * @param url    an absolute URL giving the base location of the image
 * @param name    the location of the image, relative to the url argument
 * @return       the image at the specified URL
 * @see          Image
 */
public Image getImage(URL url, String name) {
    try {
        return getImage(new URL(url, name));
    } catch (MalformedURLException e) {
        return null;
    }
}

```

Figura 1: Um pequeno programa Java com anotações JavaDoc.

Assim, utilizando o lex, poderá fazer um programa que processa apenas o código Javadoc, nomeadamente os comandos a la HTML (por exemplo, <p> para mudar de parágrafo), e que ignore a parte em Java:

```
%X JAVADOC
%%
"/**"          {BEGIN JAVADOC;}
<JAVADOC>"@param".* {printf("\nParametro: %s", &yytext[6]);}
<JAVADOC>"@return".* {printf("\nDevolve: %s", &yytext[7]);}
<JAVADOC>"@see".*    ; //ignorar
<JAVADOC>"* ".*      ;
<JAVADOC><p>          {printf("\n");}
<JAVADOC>"*/"        {BEGIN 0; printf("\n");} //regressar ao normal
<JAVADOC>\n          ;
<JAVADOC>.           ECHO;
.                   ; //caso seja codigo Java
\n                  ; //nao faz nada
%%
int main()
{
    yylex(); return 0;
}
int yywrap() {
    return 1;
}
```

Quando se volta ao *estado inicial*, o que é feito quando a instrução **BEGIN 0** é executada, o comportamento corresponde aos casos que não têm nenhum start state associado.

Note-se as 3 últimas regras desta especificação. Caso se esteja num contexto de Javadoc, o programa copia para o output o caracter encontrado. Caso contrário, o caracter encontrado é ignorado.

2 Exercícios

Exercício 1 *A conhecida linguagem de edição \LaTeX (lê-se latec), essencialmente dedicada à elaboração de artigos e relatórios científicos, e a situações onde se pretende alta qualidade gráfica em texto e fórmulas, consiste num conjunto de comandos que definem como deve aparecer o texto no resultado final. Cada comando é sempre precedido do caracter \.*

Um documento em \LaTeX deve começar sempre pelo preâmbulo, que começa com a instrução `\documentclass[opcoes]{<Classe do Documento>}`, algumas chamadas a packages (comandos do tipo `\usepackage...`) até ao início do texto propriamente dito. O texto consiste

num bloco delimitado pelo comandos `\begin{document}` e `\end{document}`. Cada bloco definido por `begin/end` é chamado ambiente e identificado entre chavetas (neste caso o ambiente é `document`).

Dentro do texto, existem vários comandos e ambientes específicos a cada situação. Neste exercício, vamos utilizar apenas os comandos `\section`, `\chapter`, `\title` e os ambientes `itemize` e `enumerate`. Em baixo, vemos um ficheiro *LaTeX* de exemplo:

```
\documentclass[onecolumn, 10pt] {article}
\usepackage{graphix}
\begin{document}
\title{O Gato}
\chapter{Introducao}
Era uma vez um gato maltes...\
\section{Seccao1}
Tocava piano e falava frances\
\section{Seccao2}
...e dava uns toques de guitarra e falava ingles tambem\
\chapter{Desenvolvimento}
Na verdade, o gato apenas:
\begin{itemize}
\item Miava
\item Ronronava
\item Fugia dos caes
\end{itemize}
\chapter{Conclusoes}
Moral da historia:
\begin{enumerate}
\item Os animais sao nossos amigos
\item Mesmo quando nao falam frances nem ingles
\item Mesmo quando nao tocam piano nem guitarra
\end{enumerate}
\end{document}
```

Um resultado possível para o processamento deste documento *LaTeX* seria a sua representação textual que se apresenta de seguida.

```
O GATO
1.  Introducao
Era uma vez um gato maltes...
1.1.  Seccao1
Tocava piano e falava frances
1.2.  Seccao2
```

...e dava uns toques de guitarra e falava ingles tambem

2. Desenvolvimento

Na verdade, o gato apenas:

- . Miava
- . Ronronava
- . Fugia dos caes

3. Conclusoes

Moral da historia:

- 1- Os animais sao nossos amigos
- 2- Mesmo quando nao falam frances nem ingles
- 3- Mesmo quando nao tocam piano nem guitarra

No sentido de ajudar os menos habituados ao \LaTeX (e que pretendem apenas compreender e extrair o conteúdo de um ficheiro `.tex`), pretende-se que construa em `lex` um mini-interpretador que produza ficheiros de texto simples com atenção aos seguintes pontos:

- *Ignore completamente o preâmbulo (ou seja, o preâmbulo não deve aparecer na saída)*
- *Coloque o título em letras maiúsculas, antecedido de um `tab \t`*
- *Elimine as indicações de `\begin{document}` e `\end{document}`*
- *Crie capítulos e secções, numerados como apresentado acima*
- *Interprete os ambientes `itemize` (colocar um `.` no início de cada linha precedida por `\item`) e `enumerate` (colocar numeração sequencial em cada `\item`)*

□

Exercício 2 *Para fazer a sua package completa, é necessária também a ferramenta que converta um texto seu em \LaTeX . Assim, poderá escrever os seus textos numa linguagem de edição inventada por si e ter o output com a qualidade \LaTeX . Na construção dessa ferramenta, atente nas seguintes regras:*

- *Garanta que a saída começa sempre com os seguintes comandos:*

```
\documentclass[a4paper,10pt]{report}
\begin{document}
```

e termina com o seguinte:

```
\end{document}
```

- *Converta sequências de maiúsculas antecedidas de um ou mais `\t` em comandos `\title{...}`*
- *Converta sequências de dígitos isolados (seguidos de ponto), por exemplo 1. em comandos `\chapter{...}`*
- *Converta sequências de pares de dígitos (intercalados com pontos), por exemplo 1.1. em comandos `\section{...}`*
- *(opcional) Converta as frases iniciadas com . ou * em comandos `\itemize`*
- *(opcional) Converta as frases iniciadas com dígito seguidas de - em comandos `\enumerate`*

□

Referências

- [1] Anexo A de Processadores de Linguagens. Rui Gustavo Crespo. IST Press. 1998
- [2] A Compact Guide to Lex & Yacc. T. Niemann.
<http://epaperpress.com/lexandyacc/epaperpress>
- [3] Manual do lex/flex em Unix (comando `man lex` na shell)