

Processamento de Linguagens (3º Ano de Mestrado Integrado em
Engenharia Informática)
Trabalho Prático 2
Relatório de Desenvolvimento

André Ricardo Covelo Germano
(A71150)

André Rodrigues Freitas
(A74619)

Sofia Manuela Gomes de Carvalho
(A76658)

23 de Abril de 2017

Resumo

Este trabalho prático tem como principais objetivos aprofundar e aplicar conhecimentos obtidos durante as aulas teóricas e práticas de Processamento de Linguagens. Além de pretender aumentar a capacidade de escrever *Expressões Regulares (ER)* para descrição de *padrões de frases* e de, através destas, desenvolver *Processadores de Linguagens Regulares* que filtrem ou transformem textos, pretende também utilizar o sistema de produção para *filtragem de texto* em *C FLEX*.

Foram escolhidos dois de seis enunciados disponíveis para este segundo trabalho prático, mais concretamente, os referentes ao Processador de Inglês corrente e ao Normalizador de Autores em BibTex.

Os resultados obtidos estão de acordo com os objetivos previamente definidos e com o que foi pedido no enunciado.

Conteúdo

1	Introdução	2
1.1	Processador de Inglês Corrente	2
1.2	Normalizador de Autores em BibTex	2
2	Análise e Especificação	4
2.1	Descrição informal do problema	4
2.1.1	Processador de Inglês corrente	4
2.1.2	Normalizador de Autores em BibTex	4
2.2	Especificação do Requisitos	4
2.2.1	Processador de Inglês corrente	4
2.2.2	Normalizador de Autores em BibTex	4
3	Conceção/desenho da Resolução	5
3.1	Algoritmos	5
3.1.1	Processador de Inglês corrente	5
3.1.2	Normalizador de Autores em BibTex	6
4	Codificação e Exemplos	9
4.1	Alternativas, Decisões e Problemas de Implementação	9
4.1.1	Processador de Inglês corrente	9
4.1.2	Normalizador de Autores em BibTex	9
4.2	Testes realizados e Resultados	10
4.2.1	Processador de Inglês corrente	10
4.2.2	Normalizador de Autores em BibTex	14
5	Conclusão	16
A	Código do Programa	17
A.0.1	Processador de Inglês corrente	17
A.0.2	Normalizador de Autores em BibTex	19

Capítulo 1

Introdução

Supervisor: André Ricardo Covelo Germano

Supervisor: André Rodrigues Freitas

Supervisor: Sofia Manuela Gomes de Carvalho

1.1 Processador de Inglês Corrente

Área: Processamento de Linguagens

O primeiro enunciado escolhido consiste na implementação de um processador de Inglês corrente. Pretende-se ler um texto corrente em inglês que faça uso das habituais contrações de vários tipos (por exemplo, *I'm, I'll, W're, can't, it's*) e reproduzir o texto de entrada na saída, expandindo todas as contrações que forem encontradas. Além disso, é pretendido, também, que se gere em HTML uma lista ordenada de todos os verbos (no infinitivo não flexionado) que se encontram no texto de entrada, tendo em conta as várias situações em que podem surgir tais como precedidos da palavra *to* ou verbos modais (*can, could*, entre outros), mas também quando são usados na forma interrogativa com o verbo *do* utilizado como auxiliar.

1.2 Normalizador de Autores em BibTex

Área: Processamento de Linguagens

O segundo enunciado tem como objetivo converter todos os caracteres com acentos explícitos em caracteres portugueses. De seguida, pretende-se reescrever todos os campos *author* e *editor* de uma lista de autores ou editores de uma publicação escrita em BibTex, mantendo os outros inalterados. Os casos em que aparecem aspas devem ser trocadas para usar sempre aspas e, além disso, deve aparecer sempre o Apelido seguido por uma vírgula e as Letras Iniciais dos restantes nomes seguidas por um ponto. Além disso, é necessário criar um grafo com todos os autores que colaboram com um dado autor.

Estrutura do Relatório

Este relatório está organizado em cinco capítulos principais que surgem após um pequeno resumo deste trabalho e são seguidos de anexos em que é apresentado todo o código desenvolvido, com todos os resultados obtidos.

O capítulo 1 consiste numa introdução em que é feita a descrição dos objetivos propostos pelo enunciado escolhido e onde se encontra a estrutura do relatório.

No capítulo 2, é inicialmente feita uma descrição informal do problema proposto, traçando as linhas gerais do mesmo e de seguida são especificados os requisitos concretos que devem ser cumpridos.

De seguida, o capítulo 3 engloba todas as opções tomadas ao longo da realização do trabalho prático, assim como as decisões que lideraram o desenho da solução e da sua implementação.

Por fim, o capítulo 5 inclui uma síntese do trabalho realizado com uma análise crítica dos resultados obtidos, possíveis melhorias e ainda trabalho futuro que poderia ser desenvolvido.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

2.1.1 Processador de Inglês corrente

O enunciado 1 tem associado a si um ficheiro anexo de um exemplo de um texto que pode ser usado para experiência ou testes, e pretende que seja desenvolvido um processador de texto com o FLEX, tal como referido anteriormente neste documento e partindo dos requisitos fundamentais que estão especificados na secção seguinte.

2.1.2 Normalizador de Autores em BibTex

O segundo enunciado escolhido tem um ficheiro anexo, o exemplos.bib, que pode ser usado para experiência ou testes. Neste enunciado também é pretendido desenvolver um processador de texto com o FLEX.

2.2 Especificação do Requisitos

2.2.1 Processador de Inglês corrente

Os requisitos fundamentais deste enunciado são:

- a) Reproduzir o texto de entrada na saída expandindo todas as contrações encontradas;
- b) Gerar em HTML uma lista ordenada de todos os verbos (no infinitivo não flexionado) encontrados.

2.2.2 Normalizador de Autores em BibTex

Os requisitos fundamentais deste enunciado são:

- a) Converter todos os caracteres com acentos explícitos em caracteres portugueses e reescrever todos os campos author e editor, mantendo os outros inalterados, substituindo as aspas por chavetas;
- b) Reescrever o ficheiro de teste para que apareça sempre o Apelido seguido por uma ”,” e que as Letras Iniciais dos restantes nomes sejam seguidas por um ”.”;
- c) Criar um grafo com todos os autores que colaboram com um dado autor.

Capítulo 3

Conceção/desenho da Resolução

3.1 Algoritmos

3.1.1 Processador de Inglês corrente

Reproduzir o texto de entrada na saída expandindo todas as contrações encontradas

Para expandir todas as contrações encontradas num texto de entrada, tivemos de especificar os padrões de frases que pretendemos encontrar no texto-fonte, recorrendo, para isso, a expressões regulares. Em relação a estas, separadas por um espaço e declaradas entre chavetas, estão as ações semânticas a realizar como reação ao reconhecimento de cada um desses padrões. Estas expressões regulares foram definidas dentro de um bloco que é definido por %%.

Para abranger todas as possibilidades de contrações num texto, definimos doze expressões regulares. As primeiras seis expressões regulares são usadas para expandir as contrações encontradas, mais especificamente, a primeira expressão e a sua ação correspondente servem para identificar abreviaturas de "I'm" e convertê-las em "I am".

As expressões seguintes funcionam de forma análoga para as outras abreviaturas consideradas, variando apenas a abreviatura específica mas não o método utilizado.

De notar que as sétima e oitava expressões regulares têm em atenção o contexto em que aparecem, sendo necessário, para isso, que a expressão regular abranja, além da contração, o texto que a sucede.

Além da definição das expressões regulares e da instrução em C a executar quando algo faz *match* com alguma expressão regular, declaramos ainda uma função *main* que permite receber um ficheiro de texto para leitura e aplicar as condições e respetivas ações definidas no bloco adequado.

O texto expandido é reproduzido no terminal ou num ficheiro para o qual se queira redirecionar, não havendo alteração do ficheiro de entrada original.

Gerar em HTML uma lista ordenada de todos os verbos (no infinitivo não flexionado) encontrados

Para gerar, em HTML, uma lista ordenada de todos os verbos (no infinitivo não flexionado) que fazem parte do ficheiro de entrada, seguimos a mesma linha de pensamento utilizada para a alínea a, sendo que neste caso foi necessário definir três variáveis globais: uma para poder identificar o ficheiro HTML onde será impressa a lista de verbos, um *array* onde serão efetivamente guardados os verbos de forma ordenada e sem repetições e um inteiro que conta o número de verbos encontrados.

As expressões regulares seguem a lógica das utilizadas na alínea a, sendo que neste caso a instrução em C a ser executada irá guardar no array referido acima apenas a parte do verbo, sendo este depois escrito entre e mais tarde na função *processa*, para permitir que seja listado.

Todas as expressões regulares declaradas e respetiva ação em C seguem este formato.

Quando o ficheiro chega ao fim (*End Of File*), é invocada a função *processa* que será responsável pela ordenação alfabética dos verbos, eliminação de verbos repetidos e impressão no ficheiro HTML de todos os campos necessários para criar uma lista ordenada e verbos no infinitivo encontrados.

Tal como para a alínea a, declaramos uma função *main* que permite receber um ficheiro de texto para leitura e aplicar as condições e respetivas ações definidas no bloco adequado.

Além desta função, são definidas mais três funções. A função *processa*, referida anteriormente, começa por fazer um *fprintf*, através do apontador recebido, que escreve no ficheiro *lista.html* o seguinte:

```
"<!DOCTYPE html>\n<html>\n<body>\n\n<h2>Lista de verbos no infinitivo</h2>\n\n<ol>\n"
```

Isto permite especificar que o tipo do ficheiro corresponde a HTML, através do campo "DOCTYPE". <html> permite indicar o começo do ficheiro HTML, <body> identifica o início do corpo do ficheiro HTML, <h2> serve para escrever o título do ficheiro, que neste caso será "Lista de verbos no infinitivo" e o permitirá iniciar a escrita de uma lista ordenada. Os campos que possuem uma / simbolizam o término da secção que antes fora aberta. De seguida, chama as funções *insertionSort* e *elimRepOrd* e imprime cada verbo armazenado. Por fim, esta função "termina" todos os campos abertos. A função *insertionSort* ordena o vetor verbos por ordem alfabética e a *elimRepOrd* permite eliminar repetições no vetor verbos previamente ordenado.

Como será possível ver mais tarde no apêndice, estas funções são chamadas antes da impressão dos verbos, como seria de esperar.

É criado, assim, um ficheiro HTML com a lista dos verbos presentes no ficheiro passado como entrada. Devemos destacar que o código desenvolvido teve em conta um ficheiro em específico (FAQ-Phyton-EN.txt) e que as exceções que foram definidas pela letra E foram aquelas que encontramos nesse ficheiro, o que significa que não há garantias de que apenas se obtenham verbos e não outras palavras se esta implementação for usada noutro ficheiro.

3.1.2 Normalizador de Autores em BibTex

Converter todos os caracteres com acentos explícitos em caracteres portugueses e Reescrever todos os campos *author* e *editor*, mantendo os outros inalterados, substituindo as aspas por chavetas

Para proceder à conversão de caracteres com acentos explícitos em caracteres portugueses, o pensamento usado foi análogo ao descrito para o enunciado 1. Contudo, neste exercício começamos por definir secções, sendo elas "AUTORES" e "TOKENS". Estas secções são definidas através de "%x".

Começamos então por converter todos os caracteres com acentos explícitos em caracteres portugueses, trocar todas as aspas por chavetas e tornar a indentação do texto mais apelativa.

De forma a facilitar a escrita de expressões regulares foram elaboradas algumas definições antes da região de %% onde serão depois usadas, nomeadamente tipos de expressões regulares. Assim, definimos um tipo de expressão regular para qualquer carácter explicitamente acentuado que precisa de ser convertido. Estas definições foram escritas, principalmente, por uma questão de comodidade, visto ser mais simples identificar dessa forma algo mais complexo e que se repete várias vezes. Foi definido um novo tipo de expressão regular para todo o tipo de acentos ou para quando se encontra um "ç" para poder fazer a conversão para os acentos explícitos em português de qualquer carácter com acentos presente no ficheiro de teste.

Este objetivo foi atingido com o uso de expressões regulares FLEX que desencadeiam ações em C. Começamos por obter todas as linhas cujos campos são *author* ou *editor* e sempre que for encontrado um desses campos são invocadas um conjunto de regras cuja *Start Condition* é *TOKENS*, regras essas que permitem efetuar o *parsing* da linha desejada e remover todos os acentos explícitos, tal como efetuar a indentação pretendida.

A primeira das regras desse género permite identificar o final da linha e, portanto, através da *Start Condition BEGIN(0)*, retornar ao estado inicial, procurando o próximo campo pretendido.

Reescrever o ficheiro para que apareça sempre o Apelido seguido por uma ",," e que as Letras Iniciais dos restantes nomes sejam seguidas por um "."

Para a segunda parte do enunciado, é requerido que um autor seja identificado pelo Apelido seguido de uma vírgula e letras iniciais dos restantes nomes seguidas por um '.'.

De forma a atingir este fim, foi necessário ter em atenção as linhas com campos *autor* e *editor*, pois poderão já haver Apelidos seguidos de vírgulas, sendo que neste caso apenas é necessário analisar e gerar os restantes nomes.

As secções definidas correspondem a AUTORES, ACASE e BCASE.

Para simplificar a escrita das expressões regulares, começamos por definir nome como sendo um P seguido de um espaço e seguido de outro P ou de um D, sendo que o espaço, o seguinte P e o D podem nunca ocorrer ou ocorrer várias vezes (nome `P([]({P} |{D}))*`). P foi definido como sendo uma letra maiúscula ou um acentM, seguido de pelo menos um lm (`([A-Z]|{acentM}){lm}+`). Por sua vez, acentM corresponde aos caracteres maiúsculos acentuados (acentM `Á|Â|É|Í|Ó|Ú`) e lm corresponde a um caracter minúsculo ou um ponto ou um acentmin (lm `[a-z]|\.|{acentmin}`). O acentmin corresponde aos caracteres minúsculos acentuados e ao c cedilhado (acentmin `á|â|ã|é|í|ó|ô|ú|ç`).

Após estas definições, passamos para as expressões regulares.

Se em qualquer contexto aparecer um author (author `(AUTHOR|author)[]*= []*`) ou um editor (editor `(EDITOR|editor)[]*= []*`), é impresso para o ficheiro author ou editor, respetivamente, e começa-se no ficheiro a secção AUTORES.

Se o ficheiro chegar ao fim, é retornado um 0.

Aquando na secção ATORES, se aparecer um nome seguido de uma vírgula, é impresso no ficheiro esse nome e começa a secção ACASE.

Se na secção ATORES aparecer uma letra maiúscula ou um acentM, ou seja, uma letra maiúscula acentuada, é copiado o caracter "0" para o sobrenome que é de seguida concatenado com o *yytext*, sendo o resultado dessa operação guardado no sobrenome. Depois, o sobrenome é concatenado com ".", ficando o resultado guardado no sobrenome. Após estas operações serem concluídas, começa a secção BCASE.

Se na secção ACASE aparecer uma letra maiúscula ou um acentM, é impresso para o ficheiro essa letra, seguida de um ponto.

Se nessa mesma secção aparecer 0 ou mais vezes lm ou um ponto, é ignorado.

Se em ACASE aparecer "and"(e) seguido de um espaço, imprime-se no ficheiro esse "and"e o espaço e começa-se a secção AUTORES.

Estando na secção ACASE e aparecendo uma chaveta direita, que determina o final de linha, começa de novo contexto indefinido.

A secção BCASE é um pouco mais complexa e, assim sendo, foi necessário recorrer à utilização de duas variáveis globais (nome e sobrenome) que permitem guardar a informação necessária para processar o nome dos autores.

Se nesta aparecer uma letra maiúscula ou um acentM ou um espaço, um D (de, da, dos ou do) e outro espaço, começa por ser concatenado um espaço para a *string* sobrenome, que é de seguida concatenado com o *yytext*. Após isto, é concatenado um ponto na *string* sobrenome. No fim desta expressão regular, sobrenome tem um espaço seguido de uma letra maiúscula ou uma letra maiúscula acentuada ou de um D e um ponto.

Por outro lado, se aparecer um P seguido de um espaço e de um "and", ou seja, quando um nome é seguido por um "and", sabemos então que existem mais nomes e portanto, para além de processar os nome e sobrenomes já guardados nas respetivas variáveis globais, o analisador léxico volta às regras da *Start Condition AUTORES*.

Quando um nome é seguido do caracter }, estamos perante o final de linha, sendo este retirado da *string yytext*. De seguida, o analisador léxico retorna novamente às regras iniciais, ou seja, a um contexto indefinido.

Qualquer outro caso nesta secção é ignorado.

No fim destas expressões, é declarada a *main*.

Criar um grafo com todos os autores que colaboram com um dado autor

Para finalizar, foi requerida a construção de um grafo com todas as colaborações entre autores. Deste modo, utilizámos quatro variáveis globais (*autores*, *colabora*, *totalColab* e *numero*).

Foram consideradas duas situações. Na primeira situação, um nome pode ser sucedido de "and", o que significa que existe um ou mais colaboradores, e a segunda situação possível é a de que um nome é sucedido por chaveta direita.

Caso um colaborador seja seguido de "and", é adicionado na variável global *autores*, na posição *numero*, que corresponde ao total de autores encontrados até ao momento.

Caso um colaborador seja seguido de }, é também adicionado à variável *autores* e é feito o processamento dessa mesma variável através da função *processa()*, que retorna um *array* de *strings* contendo todas as combinações (colaborações) entre os autores recebidos como argumento, combinações essas que através da função *insSRepetidos()* vão ser inseridas, sem repetições, na variável *colabora*. Para além disso, é também calculado o número total de colaborações entre autores.

Finalmente, quando o analisador léxico chega ao final do ficheiro são imprimidas num ficheiro com a extensão *.dot*

todas as colaborações entre autores, através da função *printGraph()*.

No *output* gerado pela função *printGraph()*, cada nó representa um colaborador e cada ligação entre nós contém o total de colaborações entre esses dois autores.

Capítulo 4

Codificação e Exemplos

4.1 Alternativas, Decisões e Problemas de Implementação

4.1.1 Processador de Inglês corrente

A implementação da resolução para o primeiro enunciado decorreu sem grandes percalços e julgamos ter obtido soluções adequadas, tendo em conta o que era possível realizar com as ferramentas pretendidas.

A alínea a não suscitou grandes dúvidas em relação ao método escolhido. No entanto, deparamo-nos com alguns casos bastante difíceis de tratar que requeriam uma análise do contexto em que surgiam, e que, por isso, não conseguimos com que o resultado obtido estivesse exatamente como pretendido. Dois exemplos concretos são os das abreviaturas *'s* e *'d*. Para a primeira das abreviaturas, pode surgir uma de três interpretações. Uma delas é a que corresponde ao *is*, quando o contexto implica a utilização do verbo *to be* no presente. No entanto, outra interpretação pode surgir facilmente quando o contexto implica a utilização do verbo *have* também no presente, correspondendo assim a abreviatura à forma verbal *has*. Outro caso ainda relativo a esta abreviatura é o caso possessivo, como, por exemplo, *"Look at Pedro's home."*, sendo que esta situação não irá constituir nenhuma das hipóteses anteriores e nem deve ser expandida, pois não é uma contração. No entanto, só através da implementação de uma exceção bastante específica que contivesse toda a expressão é que seria possível não expandir. A abreviatura *'d* pode surgir tanto quando se quer usar *had* ou *would* e deve, por isso, também ser analisado o contexto quando se quer tomar a decisão de a expandir. Devido a estas situações, optamos por usar um ficheiro de teste chamado `exemplo1.txt` que é bastante curto e, por isso, permite identificar melhor as contrações e, assim, nos casos duvidosos, escolhemos fazer a expansão que se adaptava melhor ao contexto.

Contudo, é de salientar que a alínea b também suscitou bastantes dúvidas, devido à vasta possibilidade de palavras precedidas por *to* que não estão na categoria de verbos. Um exemplo para ilustrar esta situação é a frase *"I can't go to school tomorrow because I'm feeling really sick."* onde, pela análise do enunciado, seria deduzido que *school* corresponde a um verbo quando a realidade não é essa. Para precaver este tipo de situações, declaramos algumas exceções. Como o número de exceções possíveis é realmente elevado, temos a noção de que nem todos os casos foram abrangidos e cobertos, até porque as exceções foram focadas em casos específicos dos testes realizados, não podendo esta resolução ser, por isso, generalizada a qualquer caso de teste que surja com a expectativa de que funcione com uma eficácia semelhante. Devemos também destacar as duas fases que esta alínea atravessou, visto que inicialmente não tínhamos optado por usar a ordem alfabética nem eliminar repetições, o que fez com fossem necessárias algumas alterações relativamente profundas posteriormente, visto que não estávamos a usar *arrays* ou funções auxiliares, mas sim a imprimir os verbos diretamente nas ações das expressões regulares.

4.1.2 Normalizador de Autores em BibTex

As principais dúvidas neste enunciado surgiram na implementação da terceira parte, em que foi necessário, por parte do grupo, alguma discussão em relação à resolução do problema dos colaboradores uma vez que para a implementação do grafo seria necessário não haver colaborações repetidas.

Caso nos deparemos com duas colaborações iguais, com a mesma ordem de autores, basta incrementar o total de

colaborações entre esses mesmos autores. Por outro lado, comparando lexicograficamente "Martini, R. ->Araújo, C." e "Araújo, C. ->Martini, R." obtemos que se tratam de colaborações diferentes, apesar de estarmos a lidar com a mesma colaboração. Decidimos, portanto, colocar os nomes dos autores das colaborações por ordem lexicográfica, ou seja, ambos os exemplos dados anteriormente teriam como resultado "Araújo, C. ->Martini, R." e deste modo conseguimos obter todas as colaborações não duplicadas e apresentar o resultado desejado.

4.2 Testes realizados e Resultados

Mostram-se a seguir alguns testes feitos e os respectivos resultados da execução das soluções desenvolvidas.

4.2.1 Processador de Inglês corrente

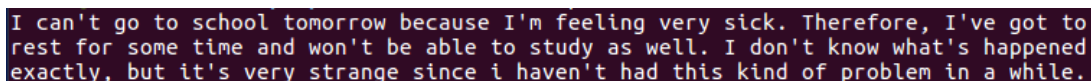
A screenshot of a text editor showing the input text for the English processor. The text is: "I can't go to school tomorrow because I'm feeling very sick. Therefore, I've got to rest for some time and won't be able to study as well. I don't know what's happened exactly, but it's very strange since i haven't had this kind of problem in a while." The text is displayed in a monospaced font on a dark background with a light border.

Figura 4.1: Texto de entrada correspondente ao ficheiro exemplo1.txt

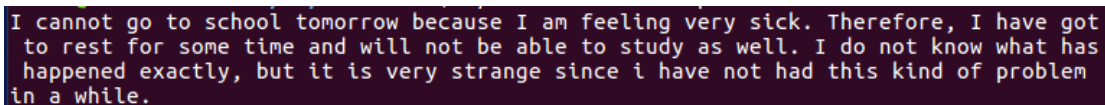
A screenshot of a text editor showing the output text from the English processor. The text is: "I cannot go to school tomorrow because I am feeling very sick. Therefore, I have got to rest for some time and will not be able to study as well. I do not know what has happened exactly, but it is very strange since i have not had this kind of problem in a while." The text is displayed in a monospaced font on a dark background with a light border.

Figura 4.2: Reprodução do texto de entrada da figura acima na saída, expandindo todas as contrações encontradas

Lista de verbos no infinitivo

1. access
2. achieve
3. add
4. allow
5. ask
6. assign
7. avoid
8. be
9. by
10. cache
11. call
12. change
13. check
14. come
15. compile
16. concatenate
17. contain
18. convert
19. create
20. decide
21. defeat
22. define
23. detect
24. do
25. download
26. emulate
27. encapsulate
28. erase
29. examine
30. find
31. force
32. form
33. get
34. give
35. grab
36. have
37. help
38. iterate
39. keep
40. know
41. lead
42. make
43. match
44. modify
45. move
46. mutate
47. override
48. pass
49. perform
50. point
51. prefer
52. print
53. produce
54. program
55. provide
56. reallocate
57. reduce
58. refer
59. remind
60. remove
61. resolve
62. result
63. return
64. run
65. save
66. say
67. search
68. see
69. set
70. share
71. solve
72. speed
73. split
74. sprinkle
75. store
76. take
77. think
78. use
79. vary
80. work
81. write

Figura 4.3: Exemplo de lista em HTML ordenada alfabeticamente sem repetições de todos os verbos (no infinitivo não flexionado) presentes no texto do ficheiro FAQ-Phyton-EN.txt

Código do ficheiro lista.html criado através da execução do código desenvolvido para a alínea b, responsável por gerar a lista de verbos visível na página anterior:

```
<!DOCTYPE html>
<html>
<body>

<h2>Lista de verbos no infinitivo</h2>

<ol>
  <li>access</li>
  <li>achieve</li>
  <li>add</li>
  <li>allow</li>
  <li>ask</li>
  <li>assign</li>
  <li>avoid</li>
  <li>be</li>
  <li>by</li>
  <li>cache</li>
  <li>call</li>
  <li>change</li>
  <li>check</li>
  <li>come</li>
  <li>compile</li>
  <li>concatenate</li>
  <li>contain</li>
  <li>convert</li>
  <li>create</li>
  <li>decide</li>
  <li>defeat</li>
  <li>define</li>
  <li>detect</li>
  <li>do</li>
  <li>download</li>
  <li>emulate</li>
  <li>encapsulate</li>
  <li>erase</li>
  <li>examine</li>
  <li>find</li>
  <li>force</li>
  <li>form</li>
  <li>get</li>
  <li>give</li>
  <li>grab</li>
  <li>have</li>
  <li>help</li>
  <li>iterate</li>
  <li>keep</li>
  <li>know</li>
  <li>lead</li>
  <li>make</li>
  <li>match</li>
  <li>modify</li>
  <li>move</li>
```

```
<li>mutate</li>
<li>override</li>
<li>pass</li>
<li>perform</li>
<li>point</li>
<li>prefer</li>
<li>print</li>
<li>produce</li>
<li>program</li>
<li>provide</li>
<li>reallocate</li>
<li>reduce</li>
<li>refer</li>
<li>remind</li>
<li>remove</li>
<li>resolve</li>
<li>result</li>
<li>return</li>
<li>run</li>
<li>save</li>
<li>say</li>
<li>search</li>
<li>see</li>
<li>set</li>
<li>share</li>
<li>solve</li>
<li>speed</li>
<li>split</li>
<li>sprinkle</li>
<li>store</li>
<li>take</li>
<li>think</li>
<li>use</li>
<li>vary</li>
<li>work</li>
<li>write</li>
</ol>

</body>
</html>
```

4.2.2 Normalizador de Autores em BibTex

Exemplo de converter todos os caracteres com acentos explícitos em caracteres portugueses:

```
author = {Ricardo Martini and Cristiana Araújo and Pedro Rangel Henriques},
editor = {Giovani Librelotto},
author = {Javier Azcurra and Mario Berón and Pedro Henriques and Maria João Varanda Pereira},
author = {Oliveira, Nuno and Henriques, Pedro Rangel and da Cruz, Daniela and Pereira, Maria João Varanda},
author = {Vilas Boas, Ismael and Oliveira, Nuno and Henriques, Pedro Rangel},
author = {Martini, Ricardo and Araújo, Cristiana and Almeida, José João and Henriques, Pedro},
editor = {Rocha, Álvaro and Correia, Maria Ana and Adeli, Hojjat and Mendonça Teixeira, Marcelo},
author = {Berón, M. and Montejano, G. and Riesco, D. and Henriques, P. R. and Debnath, N.},
```

Exemplo de reescrever todos os campos author e editor, mantendo os outros inalterados, seguindo sempre a mesma norma em que são sempre usadas as chavetas (trocando os casos em que aparecem aspas) e aparece sempre o Apelido seguido por uma “,” e as Letras Iniciais dos restantes nomes seguidas por um “.”:

```
author = {Martini, R. and Araújo, C. and Henriques, P.},
editor = {Librelotto, G.},
author = {Azcurra, J. and Berón, M. and Henriques, P. and Pereira, M.},
author = {Oliveira, and Henriques, and da Cruz, and Pereira, },
author = {Vilas Boas, and Oliveira, and Henriques, },
author = {Martini, and Araújo, and Almeida, and Henriques, },
editor = {Rocha, Á. and Correia, and Adeli, and Mendonça Teixeira, },
author = {Berón, M. and Montejano, G. and Riesco, D. and Henriques, P. R. and Debnath, N.},
```

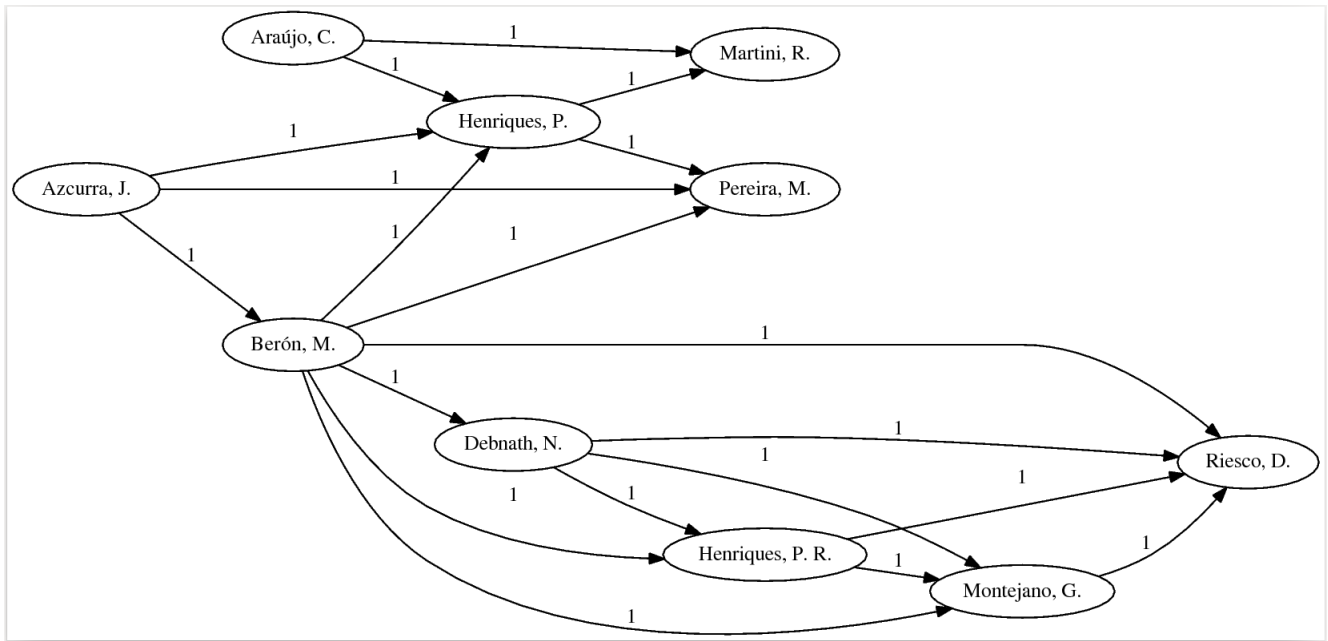



Figura 4.4: Exemplo de grafo produzido usando a linguagem Dotty e o processador dot

Capítulo 5

Conclusão

Em suma, este projeto permitiu-nos aprofundar os nossos conhecimentos na área do Processamento de Linguagens obtidos durante as aulas, especialmente os da ferramenta de processamento de texto FLEX.

Julgamos ter conseguido cumprir os objetivos definidos no enunciado e até com um pouco mais de profundidade, uma vez que desenvolvemos dois dos seis enunciados propostos, apesar de termos atravessado algumas dificuldades naturais da aprendizagem e aplicação de conceitos recentes.

Os maiores obstáculos foram a enorme variedade de possibilidades para a gramática inglesa, no enunciado 1, e a construção do grafo para o enunciado 2.

Apêndice A

Código do Programa

Mostra-se a seguir o código desenvolvido para as várias alíneas e que está explicado na secção de Algoritmos do capítulo 3.

A.0.1 Processador de Inglês corrente

3.1.1 Reproduzir o texto de entrada na saída expandindo todas as contrações encontradas

```
%option noyywrap
%%

'm          {printf(" am");}
'll         {printf(" will");}
're         {printf(" are");}
's          {printf(" is");}
've         {printf(" have");}
'd          {printf(" had");}

's[ ]got    {printf(" has got");}
's[ ]happened {printf(" has happened");}

can't       {printf("cannot");}
shan't      {printf("shall not");}
won't       {printf("will not");}

n't         {printf(" not");}

%%

int main(int argc, char* argv[]){
    if (argc == 2)
        yyin = fopen(argv[1],"r");
    yylex();
    return 0;
}
```

3.1.1 Gerar em HTML uma lista ordenada de todos os verbos (no infinitivo não flexionado) encontrados

```
%option noyywrap
%{
```

```

#include <string.h>

void processa(char* verbos[], FILE* lista, int n);
void insertionSort(char* verbos[], int n);
void elimRepOrd(char* verbos[], int n);

%}

V [Tt]o|can|could|shall|should|will|would|may|might|do|does|did
E also|the|that|lambdas|not|one|two|it|this|then|never|your|different|explicitly|uppercase|you|its|manually|

%%

FILE* lista = fopen("lista.html", "w");
char* verbos[1024];
int n = 0;

[ ]{V}[ ]{E}[ ]      {}

[ ] [Tt]o[ ] [a-z][a-z]+[ ]      {yytext[yyteng-1]='\0';verbos[n]=strdup(yytext+4);n++;}

[ ] can[ ] [a-z][a-z]+[ ]      {yytext[yyteng-1]='\0';verbos[n]=strdup(yytext+5);n++;}
[ ] could[ ] [a-z][a-z]+[ ]      {yytext[yyteng-1]='\0';verbos[n]=strdup(yytext+7);n++;}
[ ] shall[ ] [a-z][a-z]+[ ]      {yytext[yyteng-1]='\0';verbos[n]=strdup(yytext+7);n++;}
[ ] should[ ] [a-z][a-z]+[ ]      {yytext[yyteng-1]='\0';verbos[n]=strdup(yytext+8);n++;}
[ ] will[ ] [a-z][a-z]+[ ]      {yytext[yyteng-1]='\0';verbos[n]=strdup(yytext+6);n++;}
[ ] would[ ] [a-z][a-z]+[ ]      {yytext[yyteng-1]='\0';verbos[n]=strdup(yytext+7);n++;}
[ ] may[ ] [a-z][a-z]+[ ]      {yytext[yyteng-1]='\0';verbos[n]=strdup(yytext+5);n++;}
[ ] might[ ] [a-z][a-z]+[ ]      {yytext[yyteng-1]='\0';verbos[n]=strdup(yytext+7);n++;}

[ ] do[ ] [a-z][a-z]+[ ]      {yytext[yyteng-1]='\0';verbos[n]=strdup(yytext+4);n++;}
[ ] does[ ] [a-z][a-z]+[ ]      {yytext[yyteng-1]='\0';verbos[n]=strdup(yytext+6);n++;}
[ ] did[ ] [a-z][a-z]+[ ]      {yytext[yyteng-1]='\0';verbos[n]=strdup(yytext+5);n++;}

<<EOF>>      {processa(verbos, lista, n); return 0;}

%%

int main(int argc, char* argv[]){
    if (argc == 2)
        yyin = fopen(argv[1], "r");
        yyout = fopen("out.txt", "w");
    yylex();
    return 0;
}

void processa(char* verbos[], FILE* lista, int n){
    fprintf(lista, "<!DOCTYPE html>\n<html>\n<body>\n\n<h2>Lista de verbos no infinitivo</h2>\n\n<ol>\n");

    insertionSort(verbos, n);
    elimRepOrd(verbos, n);

    for(int i=0; verbos[i]!='\0'; i++){

```

```

        fprintf(lista," <li>%s</li>\n",verbos[i]);
    }

    fprintf(lista, "</ol>\n\n</body>\n</html>"); fclose(lista);
}

void insertionSort(char* verbos[], int n){
    int i, j;
    char key[1024];
    for (j=1 ; j<n ; j++) {
        strcpy(key,verbos[j]);
        i = j-1;
        while (i>=0 && strcmp(verbos[i],key)>0) {
            verbos[i+1]=strdup(verbos[i]);
            i--;
        }
        strcpy(verbos[i+1],key);
    }
}

void elimRepOrd(char* verbos[], int n){
    int i, e=0;
    for(i=0; i<n-1; i++){
        if(strcmp(verbos[i],verbos[i+1])>0 || strcmp(verbos[i],verbos[i+1])<0){
            verbos[e]=strdup(verbos[i]);
            e++;
            verbos[e]=strdup(verbos[i+1]);
        }
    }
    verbos[e+1]='\0';
}

```

A.0.2 Normalizador de Autores em BibTex

3.1.2 Converter todos os caracteres com acentos explícitos em caracteres portugueses

```

%option noyywrap
%x AUTORES TOKENS
%{
%}

a1      \{\}\ 'a\}
a2      \{\}\ 'a\}
a3      \{\}\ ~a\}
a4      \{\}\ 'A\}
a5      \{\}\ 'A\}
e1      \{\}\ 'e\}
e2      \{\}\ 'E\}
i1      \{\}\ 'i\}
i2      \{\}\ 'I\}
o1      \{\}\ 'o\}
o2      \{\}\ ~o\}
o3      \{\}\ 'O\}
u1      \{\}\ 'u\}
u2      \{\}\ 'U\}

```

```

c          \{\c\{c\}\}

D          de|da|dos|do
lm  [a-z]|\.|{\acent}|{pt}
P          ([A-Z]|{\acentM}|{pt}){lm}+
nome       {P}([ ]({P}|{D})) *
author     (AUTHOR|author)[ ]*=[ ]*
editor     (EDITOR|editor)[ ]*=[ ]*
pt         {a1}|{a2}|{a3}|{a4}|{a5}|{e1}|{e2}|{i1}|{i2}|{o1}|{o2}|{o3}|{u1}|{u2}|{c}
acent      á|à|ã|Á|Â|É|Ê|Í|Î|Ó|Ô|Ú|Û|ç
acentM     Á|Â|É|Ê|Í|Ô|Ú
char       [a-zA-z]|{\acent}|\.

```

```
%%
```

```

<*>{author}(\{|\") {BEGIN TOKENS;fprintf(yyout,"%s","author = {");}
<*>{editor}(\{|\") {BEGIN TOKENS;fprintf(yyout,"%s","editor = {");}
<*><<EOF>> {return 0;}

```

```

<TOKENS>(\{|\")\, {fprintf(yyout,"%s",",");BEGIN(0);}
<TOKENS>{a1} {fprintf(yyout,"%s","á");}
<TOKENS>{a2} {fprintf(yyout,"%s","à");}
<TOKENS>{a3} {fprintf(yyout,"%s","ã");}
<TOKENS>{a4} {fprintf(yyout,"%s","Á");}
<TOKENS>{a5} {fprintf(yyout,"%s","Â");}
<TOKENS>{e1} {fprintf(yyout,"%s","é");}
<TOKENS>{e2} {fprintf(yyout,"%s","Ê");}
<TOKENS>{i1} {fprintf(yyout,"%s","í");}
<TOKENS>{i2} {fprintf(yyout,"%s","Î");}
<TOKENS>{o1} {fprintf(yyout,"%s","ó");}
<TOKENS>{o2} {fprintf(yyout,"%s","ô");}
<TOKENS>{o3} {fprintf(yyout,"%s","Ó");}
<TOKENS>{u1} {fprintf(yyout,"%s","ú");}
<TOKENS>{u2} {fprintf(yyout,"%s","Û");}
<TOKENS>{c} {fprintf(yyout,"%s","ç");}

```

```
%%
```

```

int main(int argc,char *argv[]){
    yyin=fopen(argv[1],"r");
    yyout=fopen("parte1.txt","w");
    yylex();
    return 0;
}

```

3.1.2 Reescrever o ficheiro para que apareça sempre o Apelido seguido por uma ”,” e que as Letras Iniciais dos restantes nomes sejam seguidas por um ”.”

```

%option noyywrap
%x AUTORES ACASE BCASE
%{
    char *nome;
    char sobrenome[64];
}%

```

```

D            de|da|dos|do
lm           [a-z]|\.|{acentmin}
P            ([A-Z]|{acentM}){lm}+
nome         {P}([ ]({P}|{D})) *
author       (AUTHOR|author)[ ]*=[ ]*
editor       (EDITOR|editor)[ ]*=[ ]*
acentmin     á|à|ã|é|í|ó|õ|ú|ç
acent        á|à|ã|Á|À|É|É|Í|Í|Ó|Ó|Ú|Ú|Ç
acentM       Á|À|É|Í|Ó|Ú

%%

<*>{author}|{editor} {BEGIN AUTORES;fprintf(yyout,"%s",yytext);}
<*><<EOF>> {return 0;}

<AUTORES>{nome}\, {BEGIN ACASE;fprintf(yyout,"%s",yytext);}
<AUTORES>[A-Z]|{acentM} {
    strcpy(sobrenome,"\\0");
    strcat(sobrenome,yytext);
    strcat(sobrenome,".");
    BEGIN BCASE;
}

<ACASE>[A-Z]|{acentM} {fprintf(yyout,"%s.",yytext);}
<ACASE>{lm}* {
}
<ACASE>and[ ] {fprintf(yyout,"%s",yytext);BEGIN AUTORES;}
<ACASE>\\} {fprintf(yyout,"%s",yytext);BEGIN(0);}

<BCASE>[A-Z]|{acentM}|([ ]{D}[ ]) {strcat(sobrenome," ");strcat(sobrenome,yytext);strcat(sobrenome,".");}
<BCASE>{P}[ ]and {
    yytext[yytlen-4]='\\0';
    nome=strdup(yytext);
    fprintf(yyout,"%s, %s and",nome,sobrenome);
    BEGIN AUTORES;
}

<BCASE>{P}\\} {
    yytext[yytlen-1]='\\0';
    nome=strdup(yytext);
    fprintf(yyout,"%s, %s}",nome,sobrenome);
    BEGIN(0);
}

<BCASE>{lm}* [ ] {
}

%%

int main(int argc,char *argv[]){
    yyin=fopen(argv[1],"r");
    yyout=fopen("parte2.txt","w");
    yylex();
    return 0;
}

```

3.1.2 Criar um grafo com todos os autores que colaboram com um dado autor

```
%option noyywrap
%x AUTORES
%{
    char *autores[1024];
    char *colabora[1024];
    int totalColab[1024];
    int numero;
    void printGraph(char *colab[],int total[]);
    char **processa(char *autor[],int numero);
    void insSRepetidos(char *colab[],char *aut[ ],int total[]);

}%

D      de|da|dos|do
lm  [a-z]|\.|{acentmin}
P      ([A-Z]|{acentM}){lm}+
nome   {P}([ ]({P}|{D})) *
abv     ([A-Z]|{acentM})\ .
author (AUTHOR|author)[ ]*=[ ]*
editor (EDITOR|editor)[ ]*=[ ]*
acentmin á|à|ã|é|í|ó|õ|ú|ç
acent   á|à|ã|Á|À|É|É|Í|Í|Ó|Ó|Ú|Ú|Ç
acentM  Á|À|É|Í|Ó|Ú

%%

<*>{author}|{editor}      {numero=0;BEGIN AUTORES;fprintf(yyout,"%s",yytext);}
<*><<EOF>>                  {printGraph(colabora,totalColab);return 0;}

<AUTORES>{nome}\,([ ]{abv})*[ ]and {
                                yytext[yy leng-4]='\0';
                                autores[numero]=strdup(yytext);
                                numero++;
                                }

<AUTORES>{nome}\,([ ]{abv})*\}    {
                                yytext[yy leng-1]='\0';
                                autores[numero]=strdup(yytext);
                                if(numero>=1){
                                    char **aut=processa(autores,numero);
                                    insSRepetidos(colabora,aut,totalColab);
                                }
                                BEGIN(0);
                                }

%%

void printGraph(char *colab[ ],int totalColab[ ]){
    FILE *graph=fopen("graph.dot","w");
    fprintf(graph,"%s","digraph Colaboradores{\n");
    fprintf (graph,"    %s;\n","size=\"15,15\"");
    fprintf (graph,"    %s;\n","rankdir=LR");
    for(int i=0;colab[i]!=NULL;i++){
        fprintf(graph,"    %s [label=\"%d\";\n",colab[i],totalColab[i]);
```



```

    }
    fprintf(graph,"%s\n","}");
    fclose(graph);
}

void insSRepetidos(char *colab[],char *aut[ ],int total[]){
    int i,j;
    for(i=0;aut[i]!=NULL;i++){
        for(j=0;colab[j]!=NULL;j++){
            if(strcmp(aut[i],colab[j])==0){
                total[i]++;
                return;
            }
        }
        colab[j]=strdup(aut[i]);
        total[j]=1;
    }
}

char **processa(char *autor[],int numero){
    char *a;
    char *b;
    char **res=(char **)malloc(128*sizeof(char*));
    int i,j;
    int pos=0;
    for(i=0;i<=numero;i++){
        for(j=i+1;j<=numero;j++){
            res[pos]=(char *)malloc(128*sizeof(char));
            b=strdup(autor[j]);
            a=strdup(autor[i]);
            if(strcmp(a,b)<1){
                strcat(res[pos],"\"");
                strcat(res[pos],a);
                strcat(res[pos],"\"");
                strcat(res[pos]," -> ");
                strcat(res[pos],"\"");
                strcat(res[pos],b);
                strcat(res[pos],"\"");
            }
            else{
                strcat(res[pos],"\"");
                strcat(res[pos],b);
                strcat(res[pos],"\"");
                strcat(res[pos]," -> ");
                strcat(res[pos],"\"");
                strcat(res[pos],a);
                strcat(res[pos],"\"");
            }
            pos++;
        }
    }
    return res;
}

```

```
int main(int argc, char *argv[]){
    yyin=fopen(argv[1], "r");
    yyout=fopen("lixo", "w");
    yylex();
    fclose(yyout);
    return 0;
}
```