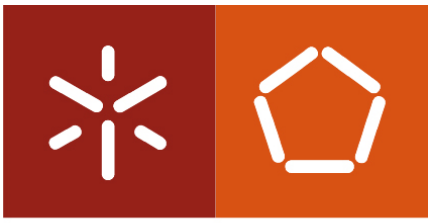

Escola de Engenharia
Universidade do Minho

DEPARTAMENTO DE INFORMÁTICA

Mestrado Integrado em Engenharia Informática

Processamento de Linguagens

TRABALHO PRÁTICO Nº 1

Normalizador de ficheiros BibTeX

Bruno Pereira
Aluno nº 72628

Ricardo Oliveira
Aluno nº 58657

Braga, 15 de Abril de 2016

Resumo

O presente documento documenta o trabalho prático em Processamento de Linguagens, relativamente a normalizadores de ficheiros BibTeX, dos quais um contador de entradas bibliográficas, que transforma essa contagem num ficheiro *HTML*, uma ferramenta de normalização para nomes de autores e normalização de campos com aspas para chavetas, bem como uma ferramente de *pretty-printing* para ficheiros BibTeX. Por último, é documentada uma ferramenta para criação de grafos em *Dot* do *GraphViz* dado um nome de um autor mostrar todas as publicações em comum, bem como respetiva densidade de publicação.

Conteúdo

Introdução

O presente documento visa a documentação do processo de aprendizagem de especificações em *Flex* e criação de filtros para diversos formatos de texto. No contexto escolhido, criações de filtros para ficheiros BibTeX. Numa primeira parte, o objetivo é a criação de uma filtro para contabilizar as entradas bibliográficas e criar um ficheiro *HTML* com o resultado. Ficheiros deste género podem crescer muito em tamanho, por vezes uma contabilização pode ajudar a construir repositórios documentais sobre determinado assunto. De igual modo, a necessidade de normalização de um documento deste é importante, porque com o passar do tempo, o acrescentar novas entradas pode criar desvios de formatação, que pode prejudicar uma pesquisa no documento pelo nome, por exemplo. Um exemplo de um normalizador deste género figura na segunda parte (capítulo 2) deste documento. Além do normalizador, na terceira parte (cap. 3), é apresentado a elaboração de um filtro para fazer um *pretty-printing* de um documento BibTeX. Por último, no quarto capítulo é apresentada uma ferramenta para criar um grafo com os coautores de determinado autor, onde os pesos das arestas são a densidade de publicação com cada coautor.

Metas e objetivos

O ambiente Linux é uma das metas deste trabalho. Existem diversas formas de fazer determinadas coisas para fazer em Linux, onde cada ferramenta tem o seu lugar e não faltam ferramentas. O domínio deste sistema operativo é importante, dado que, um programador, com conhecimento suficiente sobre este sistema operativo, tem uma liberdade que falta a outros. De igual modo, dado que as ações das especificações no *Flex* são programadas em linguagem C, um dos objetivos passa por refinar e aumentar o conhecimento da linguagem, bem como aprofundar o conhecimento em estruturas mais complexas. De igual modo, algoritmos e complexidade são serão revisitados, e a otimização será algo importante durante a elaboração do trabalho. Pretende-se, portanto, soluções eficientes. Por último, o grande objetivo é desenvolver a capacidade de escrita de *ER's* e entender como o funcionamento de processadores de linguagens regulares funcionam, usando geradores de filtros de texto como o *Flex*. Deste último objetivo será testemunha este documento.

Estrutura do Relatório

O relatório está organizado em 4 capítulos: o primeiro capítulo é referente à alínea *a*, o segundo e terceiros capítulos correspondem à alínea *b* e o quarto capitulo corresponde à alínea *c*. Cada capítulo possui três secções: *Análise do Problema*, *Desenho e implementação da solução* e *Testes e Resultados*. Na secção *Análise do Problema* expõe-se informalmente o problema, contendo conteúdo referentes aos dados, relações e possíveis abordagens à solução. Na secção *Desenho e implementação da solução* descrevem-se as especificações e respetivas ações do *Flex*, *START CONDITION*, estruturas de dados e algoritmos e funções importantes. Em seguida a secção *Testes e Resultados* apresenta os

resultados e acrescenta uma breve discussão sobre problemas de implementação, alternativas e sugestões. Note-se que neste documento, para mostrar a extensibilidade dos testes, muitos deles estão no *Apêndice*. O documento encerra com a *Conclusão* onde se avaliará a completude dos objetivos, bem como observações aos resultados obtidos.

Capítulo 1

Contagens de categorias de um ficheiro BIBTEX

1.1 Análise do Problema

Neste primeiro problema, pretende-se analisar um documento BIBTEX e fazer as contagens das respetivas categorias, tais como artigos, teses de mestrado, manuais, etc. O resultado tem que constar num ficheiro `HTML`.

1.1.1 Especificação dos requisitos

Existem pelo menos 14 tipos de entradas bibliográficas no BIBTEX, podendo haver algumas extensões ou pacotes que possuam outras — como por exemplo o BIBLATEX. Note-se que, a distinção entre o formato de ficheiro BIBTEX e o programa BIBTEX é importante. O pacote BIBLATEX pode ser usado tanto pelo programa BIBTEX como não, uma vez que o *backend* por defeito do BIBLATEX — o *biber* — suporta o formato de ficheiro BIBTEX (`.bib`). Assim, os utilizadores do BIBLATEX podem usar o mesmo ficheiro `.bib` com poucas alterações. Em consequência pode-se afirmar, que em cada ficheiro `.bib`, todos os tipos de entrada começam com , ora usam o BIBTEX, ora usem o BIBLATEX.

1.1.2 Dados

Os 14 tipos de entradas bibliográficas do BIBTEX são:

- **Article** Um artigo de um jornal ou revista.
- **Booklet** Um livro não publicado por uma editora, mas que é impresso e encadernado.
- **Book** Um livro publicado por uma editora.
- **Conference** O mesmo que *Inproceedings*.
- **Inbook** Uma parte de um livro, o qual pode ser um capítulo (ou secção ou outro qualquer) e/ou uma série de páginas.
- **Incollection** Uma parte de um livro que tem o seu próprio título.
- **Inproceedings** Um artigo de uma coleção de *papers* académicos de uma conferência.
- **Manual** Documentação técnica.

- **Mastersthesis** Uma tese de mestrado.
- **Misc** Qualquer outro documento que não se enquadre em nenhuma catgoria.
- **Phdthesis** Uma tese de doutoramento.
- **Proceedings** Coleção de *papers* académicos de uma conferência.
- **Techreport** Um relatório publicado por uma escola ou outra instituição.
- **Unpublished** Um documento com um autor e título, mas não formalmente publicado.

Para além destes tipo de entradas existem também as entradas `@STRING`, `@PREAMBLE` e `@COMMENT`, onde a primeira serve para definir abreviaturas para serem usadas no ficheiro BibTeX, a segundá define como texto especial deve ser formatado, e a última, serve para incluir comentários que não devem ser tidos em conta pelo BibTeX.

1.2 Desenho e implementação da solução

1.2.1 Expressões Regulares

Antes de se iniciar a descrição, note-se que as *ERs* estão ordenadas de forma a não haver ambiguidade.

Uma entrada de um ficheiro BibTeX começa sempre com `@`. De igual modo, os nomes de tipos de entrada podem ser escritos com maiúsculas ou minúsculas, bem como podem começar por uma maiúscula, seguidas de minúsculas. Em suma, não é *case sensitive*. Assim, na especificação do *Flex*, podemos definir uma entrada como um conjunto de caracteres, que começa com `@` seguido de uma ou mais ocorrências de caracteres, maiúsculos ou minúsculos.

Todavia, é necessário especializar a especificação, relativamente às entradas `@STRING`, `@PREAMBLE` e `@COMMENT`. Também, para os tipos de entrada bibliográficos é necessário especializar a expressão regular para os tipos comuns de entradas descritos na secção anterior. A razão desta última especialização justifica-se apenas por motivos de otimização e eficiência do filtro, que serão respondidos nas secções seguintes.

Assim, temos as seguintes *ER's*:

- Uma expressão regular para capturar ou `@STRING`, `@PREAMBLE` e `@COMMENT`, da seguinte forma:

```
1 \@[Ss][Tt][Rr][Ii][Nn][Gg]
2 \@[Pp][Rr][Ee][Aa][Mm][Bb][Ll][Ee]
3 \@[Cc][Oo][Mm][Mm][Ee][Nn][Tt]
```

  A ação nestas expressões regulares é para ignorar.[1]

- Uma expressão regular para capturar uma entrada específica, por exemplo:

```
1 \@[Aa][Rr][Tt][Ii][Cc][Ll][Ee]
```

  para capturar uma ocorrência de `ARTICLE`.

  A ação é contabilizar a ocorrência. Para a contabilização de *ER's* deste género, usou-se uma vetor de inteiros, de tamanho 14, em que cada posição corresponde a um tipo de entrada.

[1] Na especificação *Flex* estão separadas.

- Uma expressão regular para capturar uma entrada genérica, por exemplo:

```
\@[A-Za-z]+
```

  onde o valor capturado é copiado a partir do caractere `@` e inserido numa tabela de *hash*, contabilizando repetições.Especificações da tabela de *hash* encontram-se na secção seguinte.

- Uma expressão regular para ignorar tudo o resto.

Por fim, existe a nuance de se fazer a travessia da tabela de *hash* se houver elementos na tabela.

1.2.2 Estruturas de dados

Escolheu-se uma tabela de *hash* dinâmica que usa o método de *chaining*, a `uthash`[2] para utilização neste problema. Esta possui uma complexidade em termos de tempo constante na adição, remoção e procura, bem como tem uma melhor gestão de memória. No entanto, embora haja tabelas mais rápidas, utilizou-se esta tabela por uma questão de conveniência, dada a simplicidade e ter *performance* aceitável para este problema.[3]

1.3 Testes e Resultados

1.3.1 Resultados

Para testar o filtro, utilizou-se o ficheiro BibTeX dado como exemplo em http://www4.di.uminho.pt/~prh/lp.bib.

O resultado em *HTML* consta no Apêndice A, na pág. 32.

O resultado após ser executado por um *browser* é o que se segue:

Article	142
Booklet	0
Book	47
Conference	0
Inbook	3
Incollection	6
Inproceedings	209
Manual	13
Master's Thesis	2
Misc	61
Phd Thesis	21
Proceedings	4
Techreport	140
Unpublished	15
proceeding	1

Figura 1.1: Resultado visto no *Firefox*

A partir do documento que adveio da URL na secção *Resultados*, podemos constar que não existem entradas que difiram do BibTeX, a não ser a entrada *proceeding*, e que não existem *Booklets*, nem *Conferences*.

[2] https://troydhanson.github.io/uthash/

[3] *Benchmarking* relativamente a outras estruturas pode ser encontrado em http://lh3lh3.users.sourceforge.net/udb.shtml.

1.3.2 Alternativas, Decisões e Problemas de Implementação

Numa primeira implementação, existia uma *trie* para guardar os tipos de entrada que não pertencessem ao BibTeX. A travessia desta estrutura é recursiva, embora linear no número de nodos, cada nodo podia ter um *array* de apontadores de tamanho 256, podendo esse *array* ter poucas posições ocupadas. De igual modo, a *trie* que estava implementada não estava otimizada e poderia ter *bugs*. Daí a escolha passar a ser uma tabela de *hash*. Esta última estrutura, à semelhança da *trie*— ignorando o tamanho da *string* que compõe a chave —, continua a ter tempo constante de inserção, e o *overhead* é menor.

À data de redação deste relatório, chegou-se a conclusão que poder-se-ia ter escolhido uma tabela de *hash* em *open adressing* ou outra estrutura otimizada, podendo assim ter um filtro mais eficiente. Futuramente, poder-se-á tentar utilizar uma estrutura diferente e efetuar mais testes.

Capítulo 2

Ferramenta de normalização de um ficheiro BIBTEX

2.1 Análise do Problema

Para além dos tipos de entrada, é necessário especificar o conteúdo da entrada, como nomes de autor, títulos de obra, editora, etc. O BIBTEX possui propriedades definidas para cada item como campo da entrada. Para esta parte do trabalho, é pedido o desenvolvimento de uma ferramenta de normalização dos nomes dos autores no campo respetivo, no formato *N. (ome) Apelido*, que de igual modo normalize todos os campos entre aspas, com chavetas.

2.1.1 Especificação dos requisitos

2.1.2 Dados

Os nomes dos autores podem ter muitos formatos. Como por exemplo:

- *Donald E. Knuth*
- *D. E. Knuth*
- *Knuth, Donald E.*
- *Knuth, PhD, Donald E.*
- *Nicollo Alighieri Franchi-Zanettachi*
- *Daniela da Cruz*

Assim, há uma necessidade de especializar um conjunto de *ER's* para tratar cada caso, com especial atenção para os nomes no formato *Apelido, Nome*.

De igual modo, temos que cada campo pode começar com uma chaveta ou aspas, terminando de igual forma, com a chaveta fechada ou aspas correspondente.

Um conceito importante no TEX em geral, é que um documento está *bem formado* se todas as chavetas abertas tiverem a chaveta fechada correspondente. De facto, existem estilos de bibliografias que convertem o primeiro caractere que compõe o valor do campo em maiúscula e os restantes em minusculas. Esta funcionalidade ocorre para nomes de um título ou outro campo, que não o do autor. Por vezes é necessário manter as maiúsculas, dado que existem valores de campo em que, por

exemplo, o primeiro caractere de cada palava está capitalizado. O BibTeX permite ao utilizador abrir e fechar chavetas em torno do conjunto de caracteres onde se pretende manter a capitalização. A relevância deste contexto será explicada na secção seguinte.

2.2 Desenho e implementação da solução

2.2.1 Expressões Regulares

Antes de se iniciar a descrição, note-se que as *ERs* estão ordenadas de forma a não haver ambiguidade.

2.2.1.1 *INITIAL* START CONDITIONS

Qualquer campo no BibTeX tem sempre o identificador do campo (*author*, *title*, etc.) seguido de um '=', começando o valor do campo com a abertura de aspas ou chavetas. Todavia, entre os o identificador do campo, '=' e `{`, pode não haver espaços, ou pode haver um ou mais espaços. Como a ferramente de normalização tem duas funcionalidades diferentes conforme os campos, é necessário ter duas *ER's*: uma que trate de tudo o que é necessário fazer com o campo autor e outra, genérica, que trate dos restantes no contexto da normalização das chavetas. Dado que, a captura do campo autor e do processamento das chavetas são dois contextos diferentes, é necessário recorrer ao uso de *START CONDITIONS*. Assim, para estes últimos implementou-se a *START CONDITION* `AUT` para tratar do autor, e a *START CONDITION* `CHAV` para tratar das chavetas dos outros campos.

Após o exposto atrás, temos duas expressões regulares, tais que:

- Captura campo autor

  ```
  [Aa][Uu][Tt][Hh][Oo][Rr][ ]*"="[ ]*[{"]
  ```

  A ação para esta expressão regular é colocar no último caractere capturado uma chaveta aberta, imprimir no *stdout* e iniciar `AUT`.

- Captura qualquer outro campo;

  ```
  [A-Za-z]+[ ]*"="[ ]*[{"]
  ```

  A ação para esta expressão regular é colocar no último caractere capturado uma chaveta aberta, imprimir no *stdout* e iniciar `CHAV`.

2.2.1.2 *AUT* START CONDITION

Nesta *START CONDITION* faz-se a distinção dos nomes, conforme na *Secção* 2.1.1. No entanto, é necessário distinguir o que é um nome de um autor. Um nome de uma autor pode-ser seguido de um *and*, se houver mais que um autor, e também pode ser único ou o último, e terminar em aspas ou numa chaveta. Dentro do nome, faz-se a distinção de um nome poder ser uma inicial, ou uma palavra que inicie com maiúscula, seguida de uma ou mais letras minúsculas — por definição, um nome próprio —, ou pode ser um nome composto (dois nomes próprios separados por um hífen). De igual modo, os nomes podem conter uma preposição dentro do nome, e podem ter vários espaços e/ou tabulações antes e depois do nome. Acresce também a condição especial de que os nomes podem estar no formato *Apelido, Nome* e neste caso é necessário uma novo contexto para tratar este caso especial. Para isso, criou-se a *START CONDITION* `PREFORMAT`.

- Captura de aspas ou uma chaveta no final campo.

```
[}"]
```

  Neste caso, como se poderá ver mais à frente, pode aparentar ser redundante. Ou seja, já existem *ER's* que tratam da chaveta ou aspas como fim de campo. Todavia, por uma questão de coerência e segurança, em caso de uma captura para entrar nos estado da *START CONDITION* `PREFORMAT`, no final do processamento do campo nesta condição, volta-se sempre ao estado anterior `AUT`. Assim, o final de campo é sempre processado no mesmo contexto.

  A ação nesta *ER* é consumir o valor, imprimindo no *stdout* uma chaveta, voltando à *START CONDITION* `INITIAL`.

- Captura de 1 ou mais espaços ou tabulações.

```
[ \t]+
```

  Esta *ER* garante que espaços ou tabulações em torno dos nomes são consumidos, de forma a ter apenas os caracteres correspondentes aos nomes. De igual modo, imprime um espaço no *stdout*.

- Captura de um espaço ou tabulação antes e depois de uma preposição, e a própria preposição.

```
[ \t][a-z]+[\t ]
```

  As preposições em nomes com *Daniela da Silva* são ignorados. A ação é mesma que na *ER* anterior.

- Captura de uma inicial ou um nome próprio, que não apelido.

```
[A-Z]((\.)?|[a-z]+)
```

  A ação neste caso é obter o primeiro caractere da captura da expressão, e imprimir no *stdout* o mesmo caractere seguido de um ponto.

- Captura de um nome próprio, que é apelido, pode ser composto e é o último da listagem ou único.

```
((-)?[A-Z][a-z]+)+[ \t]*[}"]
```

  O nome pode ter ou não iniciar com um hífen, seguido do nome próprio. Estes dois itens podem ocorrer uma ou mais vezes. Por exemplo, no apelido do nome *Nicollo Franchi-Zanettachi*. Neste caso, o nome do autor é o último nome listado ou o único. Assim, espaços e tabulações antes do final da linha são capturados, 0 ou mais vezes (apelido pode estar junto das aspas ou da chaveta).

  A ação pretendida aqui é modificar o último caractere para uma chaveta, imprimir o resultado para o *stdout* e voltar a *START CONDITION* `INITIAL`.

- Captura de um nome próprio, que é apelido, pode ser composto mas não é o único na listagem de autores.

```
((-)?[A-Z][a-z]+)+[ \t]+(and)[ \t]+
```

  A ação correspondente é em quase tudo semelhante à ação anterior, no entanto, captura todos os espaços e tabulações, antes e depois do separador *and*, e o separador *and*. Neste caso imprime para o *stdout* a captura. Assim o próximo fica livre de espaços no inicio.

- Captura de uma linha com a formatação *Apelido, Nome*.

```
((-)?[A-Z][a-z]+)+[,]+[ \t]
```

  A *ER* captura, a além do nome, pelo menos uma vírgula depois do nome, estando este seguido de espaços. Assume-se que, se a formatação no primeiro nome possui vírgulas, logo a restante está na mesma formatação. A ação é colocar o apontador de leitura do *Flex* no inicio da linha, colocar uma variável inteira para um índice de um vetor a 0, inicializar *array* de *strings* a `NULL`[1]. Em seguida inicia a *START CONDITION* `PREFORMAT`.

- Captura tudo o resto incluindo *newline*.

```
(.|\n)
```

  A ação é ignorar tudo, que seja capturado por esta *ER*.

2.2.1.3 *PREFORMAT* START CONDITION

Esta *START CONDITION* tem as *ER's* iguais, exceto a *ER* `[, \t]+` e o separador *and* tem um tratamento diferente, bem como o final do valor do campo. A maior parte das ações foi redefinida para para este novo contexto.

- Captura de aspas ou uma chaveta no final campo.

```
[}"]
```

  Como foi anteriormente mencionado, quando capturado ou aspas ou uma chaveta, obriga-se o *Flex* a colocar a captura no *stdin* (`yyless(0)`). De seguida coloca-se o índice do *array* de *strings*, mencionado anteriormente, na posição inicial e executa-se uma função para imprimir os valores contidos no *array*.[2]

- Captura de 1 ou mais espaços, tabulações ou vírgulas.

```
[, \t]+|
```

[1] Algoritmo e código será apresentado na *Subsecção 2.2.2 na pág. 15*
[2] Algoritmo e código descrito na *Subsecção 2.2.2, na pág. 15*

A semelhança da anterior *ER*, para além de garantir que espaços ou tabulações em torno dos nomes são consumidos, garante de igual modo que vírgulas sejam consumidas. No entanto, não imprime um espaço. No seu lugar da ação de impressão de um espaço, a variável do tipo inteiro para o índice do *array* de *strings* é incrementada por uma unidade. Ou seja, por cada nova ocorrência de um novo nome, o índice do *array* já se encontra na posição correta.

- Captura de um espaço ou tabulação antes e depois de uma preposição, e a própria preposição.

```
[ \t]+(and)+[\t ]+
```

A ação definida captura o separador *and*, caso exista mais que um autor, após a ocorrência deste. Note-se que pode encontrar um ou mais mais espaços, bem como tabulações antes e depois deste separador. Além disso, coloca o índice do apontador do *array* de *strings* auxiliar na posição inicial, imprimindo os nomes próprios do autor pela ordem desejada — *N. Apelido* em troca de *Apelido, Nome*. Após este passo, imprime no *stdout* a *string and* com um espaço de cada lado, iniciando a *START CONDITION* `AUT`. Note-se que, o tratamento do conjunto de nomes é aqui diferente da *START CONDITION* `AUT`, uma vez que, em `AUT`, não se podia saber qual seria o último nome, a não ser uma implementação especializada na parte da ação em linguagem *C*. O intuito deste projeto foi sempre usar ao máximo a funcionalidade do *Flex*, e tentar usar ao máximo *ER's*. Aqui, como sabemos que o último nome do autor é logo o primeiro a ser listado.,

- Captura de um espaço ou tabulação antes e depois de uma preposição, e a própria preposição.

```
[ \t][a-z]+[\t ]
```

A ação é mesma que na *ER* equivalente, descrita na *Subsubsecção* 2.2.1.2 na pág. 10, que é imediatamente antes desta.

- Captura de uma inicial ou um nome próprio, que pode ser composto.

```
((-)?[A-Z]((\.)|[a-z]+))
```

A *ER* tem o mesmo intuito que a *ER* equivalente da *Subsubsecção* 2.2.1.2 na pág. 10, no âmbito da captura. No entanto, a ação é diferente. Esta passa por comparar a posição atual do índice do *array* auxiliar para identificar a ordem dos nomes. Assim, para cada ocorrência de um nome próprio de um autor, se for diferente da posição 0, é colocado no valor de `yytext` na posição 1 um um ponto e na posição seguinte, o caractere `'\0'`, copiando esse valor para o *array* de *strings* auxiliar, na posição atual do índice do *array*. Caso contrário, copia o valor completo da *string* para a posição 0, do mesmo *array* auxiliar.

- Captura tudo o resto incluindo *newline*.

```
(.|\n)
```

A ação é ignorar tudo o que seja capturado por esta *ER*.

2.2.1.4 *CHAV* START CONDITION

No contexto `CHAV`, apenas se faz a captura das aspas ou chavetas de todos os outros campos. As chavetas ou aspas do campo do autor já estão previstas no devido contexto. No entanto, aqui pode ocorrer o novo contexto, já mencionado na secção *Análise do Problema* deste capítulo. Deste modo, há a necessidade de se ter uma nova *START CONDITION* `SPEC`.

- Captura uma chaveta aberta.

  ```
  [{]
  ```

  A ação consiste imprimir para o *stdout* a chaveta e iniciar a *START CONDITION* `SPEC`.

- Captura do fim de campo, podendo ser uma chaveta ou aspas.

  ```
  [}"]
  ```

  Ação: imprimir a chaveta de fecho, e voltar para as *START CONDITION* `INITIAL`

- Captura qualquer outro campo;

  ```
  (.|\n)
  ```

Ação: imprimir tudo o resto, incluindo *newlines* para *stdout*.

2.2.1.5 *SPEC* START CONDITION

Há apenas acrescentar sobre esta *START CONDITION*, que apenas é um *workaroud* para evitar de capturar uma chaveta de a meio do valor do campo, e ser passível de ser considerada fim do valor de campo.

- Captura o fecho de chavetas.

  ```
  [}]
  ```

  Imprime para o *stdout* a mesma chaveta.

- Captura qualquer outro campo;

  ```
  (.|\n)
  ```

  Imprime tudo o resto no *stdout*.

2.2.2 Algoritmos

```
void print_array (char ** array)
{
    int i;

    for (i = 1; i < ARRAY_SIZE&&array[i]; i++)
        {

              printf("%s ", array[i]);
        }
    printf("%s", array[0]);


}
```

A função acima corresponde à função de impressão dos nomes próprios dos autores na *START CONDITION* `PREFORMAT`. Note-se que apenas valores existentes no *array* são imprimidos no *stdout*, começando pela segunda posição. A primeira posição é imprimida no fim.

```
void clean_array (char ** array)
{
    int i;

    for (i = 0; i < ARRAY_SIZE&&array[i];
               free(array[i]), array[i++]=NULL);


}
```

A função acima corresponde à função de inicialização do *array* dos nomes próprios dos autores na *START CONDITION* `PREFORMAT`. Note-se que liberta a memória e inicializa valores previamente existentes.

2.3 Testes e Resultados

2.3.1 Resultados

O ficheiro de usado para este teste pode ser visto no *Apêndice* B na pág. 34.
O resultado pode ser consultado no *Apêndice* C na pág. 39

2.3.2 Alternativas, Decisões e Problemas de Implementação

Adicionalmente à solução descrita neste capitulo, poder-se-ia ter implementado ou mais uma *START CONDITION* ou possivelmente mais algumas *ER's* que tratassem de nomes de sufixo como em *Knuth, PhD, Donald E.*.

Assumiu-se uma codificação *ASCII*, pelo que não foram tratados caracteres em *UTF-8* ou *ISO 8859-1*. Para tal ter-se-ia que tratar os caracteres com o tamanho de dois *bytes* e capturar sequências de escape para caracteres especiais em determinado ficheiro BibTeX e guardá-los como caracteres de dois *bytes*.

Capítulo 3

Ferramenta *pretty-printing* de um ficheiro BIBTEX

3.1 Análise do Problema

Nesta parte, o desafio é elaborar uma ferramenta de *pretty-printing* que indente corretamente cada campo, escreva um autor por linha e coloque sempre no início os campos autor e título.

3.2 Especificação dos requisitos

3.2.1 Dados

No BIBTEX, cada tipo de entrada bibliográfica tem campos opcionais e obrigatórios. Na resolução deste problema apenas se centrou nos obrigatórios, uma vez que, se se quiser uma ferramenta genérica que inclua outros pacotes, o número de campos é elevado. De igual modo, os campos obrigatórios podem ser opcionais para algum tipo de entrada, e vice-versa. Em extensões, como no BIBLATEX, alguns campos são comuns a outros campos, por uma questão de compatibilidade. Por exemplo, campo autor e título são comuns à maioria dos tipos de entrada, no caso da entrada `MISC` são opcionais. Omitiu-se, de igual modo, os campos `note`, `abstract` entre outros, que podem ter demasiada informação, para um *pretty-printing*.

Os campos considerados foram:

- Organização
- Forma de publicação[1]
- Instituição
- Publicação
- Titulo do livro
- Jornal
- Edição
- Capitulo

[1] *How Published*

- Morada
- Volume
- Serie
- Escola
- Numero
- Editor
- Autor
- Titulo
- Págs
- Mês
- Ano
- Tipo

Note-se que estes campos podem ser tanto obrigatórios como opcionais. A ordem com que os campos estão colocados não tem a obrigatoriedade de uma sequência. Por exemplo o campo autor pode ser colocado no final da enumeração dos campos, sem nenhum efeito colateral. O que decide a ordem dos elementos é sempre o estilo de bibliografia.

3.3 Desenho e implementação da solução

A sugestão de solução apresentada, possui quatro *START CONDITIONS*: a *SC* `ENTRY`, *SC* `AUT`, a *SC* `FIELD` e, por último, a `SPEC`. A necessidade de ter as primeiras *SC* deve-se à necessidade de criar um contexto para tratar os autores, bem como o título, sendo estas ativadas dentro *SC* `ENTRY`. O intuito *SC* `SPEC` já foi descrita em secções anteriores — capturar par de chavetas, para evitar conflitos com as outras *ER's*. De igual modo, usaram-se três variáveis globais do tipo inteiro: uma para guardar o estado das *SC* no caso da `SPEC`, dois contadores, um para um índice de uma *string*, outro para guardar o índice de um *array* de *strings* bi-dimensional. Além, destas variáveis de valor inteiro, criaram-se, como já mencionado, uma *string* e um *array* bi-dimensional para guardar o resultado das ocorrências dos campos. Criou-se este último, com o intuito de guardar nas primeiras posições os valores do campos autor e título, seguido de tudo o resto. Assim, após a captura do fim da entrada bibliográfica, o resultado é passado ao *stdout* pela ordem correta, iniciliazidando o índice do *array* bi-dimensional, de cada vez que é encontrada uma entrada bibliográfica. Pr último, o *array* é bi-dimensional dado que se pretende guardar na primeira linha a legenda do campo e na segunda o valor do campo. A especificação de cada *SC*, que implementa estes conceitos segue-se nas seguintes secções. Note-se que, que todas a *START CONDITIONS* estão ordenadas dentro do seu contexto, de forma a não haver ambiguidade.

3.3.1 Expressões Regulares

3.3.1.1 *START CONDITION* `INITIAL`

- Captura o início de uma entrada, ou seja, um `@` seguido de uma ou mais caracteres maiúsculos ou minúsculos.

  ```
  \@[A-Za-z]+
  ```

  Como já foi anteriormente mencionado, a captura deste valor através da *ER* descrita, despoleta uma ação, tal que a posição do *array* bi-dimensional é inciada logo na terceira posição, uma vez as duas primeiras estão reservadas para o autor e o título. De igual modo, inicia a *SC* `ENTRY`, imprimindo no *stdout* uma pequeno separador composto por cardinais.

- Captura tudo o resto, incluindo o caractere *newline*.

  ```
  (.|\n)
  ```

  A ação é ignorar tudo o que é capturado por esta *ER*.

3.3.1.2 *START CONDITION* `ENTRY`

A *SC* `ENTRY` está no contexto da entrada bibliográfica, onde os vários campos são capturados.

- Captura chaveta correspondente ao final da entrada bibliográfica.

  ```
  [}]
  ```

  Aqui é invocada a função de impressão dos campos pela ordem desejada, voltando a *SC* `INITIAL`.

- Captura uma chaveta seguida de 0 ou mais espaços, com uma vírgula a seguir.

  ```
  [}][ ]*[,]
  ```

  A ação aqui definida é ignorar a captura. Este *ER* serve de artifício para não haver ambiguidade com o final dos campos, dado que a expressão é maior. Note-se que, caso não se recorresse a este artifício, a captura dos campos poderia terminar a meio.

- Captura o campo *Organization*, *case-insensitive*, seguido 0 ou mais espaços, um *"="*, com 0 ou mais espaços de seguida, podendo terminar ou não numa chaveta ou aspas.

  ```
  [Oo][Rr][Gg][Aa][Nn][Ii][Zz][Aa][Tt][Ii][Oo][Nn][ ]*"="[ ]*[{"]?
  ```

  Após a captura, o índice da *string* auxiliar, para armazenar temporariamente o valor dos caracteres capturados, é inicializado, sendo guardada a legenda do campo na primeira linha na coluna correspondente à posição da ocorrência. Em seguida é inicializada a *SC* `FIELD`, correspondente ao contexto de manipulação de qualquer outro campo, que não o autor e o título.

- Captura similar a anterior, exceto a aspas ou chavetas de início do valor do campo são obrigatórias. Neste caso a captura é do campo autor.

```
[Aa][Uu][Tt][Hh][Oo][Rr][ ]*"="[ ]*[{"]
```

A ação é igual à de *ER* anterior, com a exceção da posição do *array* que é utilizada diretamente, e é iniciada a *SC* `AUT`.

- Captura igual à anterior, só que neste caso, a captura é do campo do título. A *SC* iniciada é a `TITLE`.

```
[Tt][Ii][Tt][Ll][Ee][ ]*"="[ ]*[{"]
```

Estas *ER's* estão aqui a título exemplar, dado que existem *ER's* para cada campo obrigatório mencionado na secção *Análise do Problema*.

- Captura tudo o resto incluindo o caractere *newline*, sendo a ação ignorar a captura.

```
(.|\n)
```

3.3.1.3 *START CONDITION* `AUT`

Nesta *SC* trata-se do campo do autor, de forma que cada autor apareça um por linha e tenha a devida indentação.

- Captura um ou mais espaços ou tabulações, seguidas do separador *and*, sendo seguidas novamente, por um ou mais espaços ou tabulações.

```
[ \t]+"and"[ \t]+
```

Nesta captura é copiado para a *string* auxiliar, uma *string* com o caractere *newline*, para colocar um autor por linha, e três tabulações e um espaço para dar a indentação pretendida. O valor do índice é incrementado pelo o número de caracteres que é copiado, ou seja incrementado por 5.

- Captura aspas ou chavetas que correspondem ao final do valor do campo do autor.

```
[}"]
```

Adiciona-se um *newline* à *string* auxiliar — incrementando por um o valor do seu índice — e o valor da *string*, já com todos os autores, é copiada para a segunda linha, na coluna 0 do *array* bi-dimensional. Após este passo volta para a *SC* que gerou a *SC* `AUT`, neste caso `ENTRY`,

- Captura tudo o resto, incluindo o caractere *newline*.

```
(.|\n)
```

Nesta captura, o valor contido na variável do *Flex*, `yytext` é copiado para a *string* auxiliar. Como a captura é caractere a caractere, copia-se apenas o primeiro caractere desta *string*. Como em anteriores adições à *string* auxiliar, a posição é incrementada.

3.3.1.4 *START CONDITION* `TITLE`

- Captura dois ou mais espaços.

  ```
  [ ]{2,}
  ```

  Apenas serve para consumir espaços a mais, imprimindo no *stdout* um espaço.

- Captura qualquer chaveta aberta, que não a do fim de campo.

  ```
  [{]
  ```

  A ação desta captura é devido ao que foi exposto em capítulos anteriores. No entanto, como foi anteriormente explicado, vários contextos utilizam a *START CONDTION* `SPEC`. Assim, como em anteriores *START CONDTIONS* o valor do estado é guardado numa variável global, com a macro `YYSTATE`. O chaveta é consumida.

- Captura o final do campo caso seja aspas ou o fecho de chavetas.

  ```
  [}"]
  ```

  A ação é copiar um caractere *newline* para a *string* auxiliar, para depois o valor desta *string* ser copiado para a segunda linha e segunda coluna do *array* bi-dimensional. Após este passo, regressa à *SC* `ENTRY`.

- Captura tudo o resto.

  ```
  (.)
  ```

  Copia o valor da ocorrência para a *string* auxiliar.

- Captura do caractere *newline*

  ```
  (\n)
  ```

  A ação é ignorar a captura. Apenas serve para consumir eliminar quebras de linha no campo do título.

3.3.1.5 *START CONDITION* `FIELD`

Na *SC* `FIELD` é processado o conteúdo de todos os outros campos mencionado na primeira secção deste capítulo, que não o autor e o título.

- Captura campo com *URL*.

  ```
  "\\url{"
  ```

  À semelhança do descrito noutros capítulos, a *tag*, começa com aspas tendo que terminar em aspas. Daí que se tenha usado o conceito da do capítulo anterior para capturar esta *tag*. A ação tem por base consumir o valor desta captura, e iniciar a *SC* `SPEC`, guardando o estado da *SC* no processo da ação.

- Captura dois ou mais espaços.

  ```
  [ ]{2,}
  ```

  O procedimento associado a esta captura é simplesmente acrescentar um espaço à *string* auxiliar, consumindo os espaços em excesso.

- Captura a abertura de chavetas.

  ```
  [{]
  ```

  À semelhança da *ER* equivalente na *SC* `TITLE`, na captura desta *ER*, guarda-se o estado da *SC* atual para iniciar a *SC* `SPEC`, pelas mesmas razões.

- Captura opcional de um fecho de chavetas ou aspas, seguido de 0 ou mais espaços, terminados por uma vírgula.

  ```
  [}"]?[ ]*[,]
  ```

  Existem campos cujo valor não está delimitado nem por aspas nem por chavetas. Assim, é necessário definir o fim do valor de um campo de outra forma, ou seja, o campo tem que terminar por vírgula. No entanto, isto nem sempre acontece, porque, caso o campo seja o último há definir outra forma de o campo acabar. Esse é o intuito da *ER* seguinte. O processo associado a esta *ER* é uma cópia de caractere *newline* para a *string* auxiliar, sendo feita a cópia da *string* auxiliar para o *array* bi-dimensional, na segunda linha, na índice equivalente à contagem das ocorrências dos campos. No fim volta para a *SC* `ENTRY`.

- Captura opcional de um fecho de chavetas ou aspas, seguido de 0 ou mais espaços, com um *newline* opcional, seguido 0 ou mais espaços, terminado por uma chaveta.

  ```
  [}"]?[ ]*(\n)?[ ]*[}]
  ```

  Esta *ER* define a outra forma de de um campo terminar. Isto é quando o campo é o último. Note-se que, a chaveta que termina esta *ER* tem um tratamento no contexto da *ER* `ENTRY` — o fim da entrada bibliográfica. Deste modo, é necessário uma manobra adicional: colocar o *Flex* na posição para reler a chaveta no contexto correto. Para isso usa-se o `yyless (yyleng-1)`. Esta função é invocada antes de voltar para *SC* `ENTRY`. O resto do processo é igual ao da especificação anterior.

- Captura tudo, exceto o caractere *newline*.

  ```
  (.)
  ```

  Ação: copiar a ocorrência do caractere da captura para *string* auxiliar.

- Captura o caractere *newline*.

  ```
  (\n)
  ```

  A operação associada é ignorar, ou seja, a *string* auxiliar não terá quebras de linha.

3.3.1.6 *START CONDITION* `SPEC`

- Captura uma chaveta aberta.

  ```
  [}]
  ```

  A ação é similar a contextos descritos em capítulos anteriores — caso o fecho de uma chaveta seja detetado neste contexto, coloca o *Flex* na posição de releitura dessa chaveta para ser processado pelo contexto correspondente. Chama-se à atenção de que se usou a variável global de estado descrita anteriormente, para iniciar um novo contexto. Isto acontece, uma vez que tanto títulos e outros campos podem ter uma abertura de chavetas no meio do valor do campo, daí do uso de uma variável global.

- Captura tudo o resto.

  ```
  (.)
  ```

  Na captura, ação é guardar o valor na *string* auxiliar, ignorando a chaveta.

- Captura o caractere *newline*.

  ```
  (\n)
  ```

  Mantém-se a intenção de não ter quebras de linhas dentro dos campos, que não sejam autor.

3.3.2 Algoritmos

A função abaixo é a função utilizada para imprimir os valores guardados no *array* bi-dimensional. A impressão vai até ao total de posições incrementadas na captura dos campos. Assim, não é necessário alocar memória para o *array* 2D.

```
void print_campos()
{
    int j;

    for(j = 0; j < pos; j++){
        printf("%s %s", fields[0][j], fields[1][j]);
      free(fields[0][j]);
      free(fields[1][j]);
      fields[0][j]=NULL;
      fields[1][j]=NULL;
    }

}
```

Na `main` todas as entradas não nulas são libertadas da memória.

```
int j;

    yylex();
 for(j = 0; j < MAX_ENTRIES; j++)
        {
            if(fields[0][j])
                free(fields[0][j]);
            else if (fields[1][j])
                free(fields[1][j]);

        }
```

3.4 Testes e Resultados

3.4.1 Resultados

Os resultados da aplicação deste filtro ao ficheiro no apêndice B na pág. 34, estão no apêndice D na pág. 42.

3.4.2 Alternativas, Decisões e Problemas de Implementação

Houve diversos problemas na implementação dado à quantidade de exceções que podem ocorrer num ficheiro BibTeX e a intenção de usar uma solução completamente "nativa"do *Flex*, isto é, apenas com *ER's* e funções do *Flex*. Logo, de forma a simplificar o problema, não se está a considerar caracteres especiais, nem escape de caracteres. Até à data de redação deste documento desconhece-se uma solução "nativa"que vá de encontro à especificação do enunciado — ordem pré-definida dos campos na apresentação. No entanto houve várias tentativas, onde uma solução em particular fazia o *pretty-printing*, no entanto, ter-se-ia que assumir que o utilizador do ficheiro BibTeX colocava os campos autor e titulo em primeiro lugar. Esta solução não vai de encontro à especificação do enunciado.

Numa melhoria futura, acrescentar suporte para caracteres especiais à semelhança de capítulos anteriores.

Capítulo 4

Grafo de associações de um autor em *Dot* de um ficheiro BIBTEX

4.1 Análise do Problema

Nesta parte, requere-se que se faça o grafo de associações para um dado autor, dos autores que usualmente publicam com ele. A linguagem a ser usada é linguagem *Dot* do *GraphViz*, para renderizar o grafo, tendo que o filtro em *Flex* gerar no *stdout* o grafo nessa linguagem.

4.2 Especificação dos requisitos

4.2.1 Dados

Em relação aos dados do problema, pouco mais há a acrescentar do que já se mencionou em capítulos anteriores, relativamente ao BIBTEX — o campo autor podes ter um ou mais autores, cada um separado por *and*, podendo o campo estar dentro de chavetas ou aspas.

Em relação ao *Dot* existem algumas considerações a ter, nomeadamente no desenho do grafo através da linguagem. Pretende-se um grafo, de preferência orientado, onde numa única página seja possível mostrar todas as associações. Cada associação pode ter um peso, correspondente à densidade de publicação e existe uma, e uma só associação entre o autor escolhido e cada um dos seus co-autores. Além do mais, podem ocorrer os seguintes casos: o autor tem vários co-autores, o autor nunca publicou com ninguém, e o autor escolhido não existe. Acresce-se que o grafo é sempre produzido, estando vazio no caso de o autor não existir ou haver apenas um nodo com o nome do autor escolhido, no caso de autor ter sempre publicado sozinho.

4.2.2 Relações

Relativamente às associações, existem relações subjacentes que é preciso interpretar. Pretendendo-se todas as associações entre cada autor que ocorre no campo homólogo no ficheiro do BIBTEX, é necessário construir um grafo com todas as associações entre todos os autores. Por exemplo, temos 2 entradas bibliográficas tal que:

- C and B

  O autor C publica com B, por sua vez, B publica com C,

- A and B and C

  O autor A publica com B e com C, e B publica com A e com C, e C publica com A e com B.

Ou seja, para uma linha com N autores, para cada autor exitem N-1 associações. Para manter a consistência do grafo, e necessário inserir essas N -1 associações mais o nome do autor, N vezes.

4.3 Desenho e implementação da solução

Na implementação usou-se para criar o grafo, uma tabela de *hash* multi-nível, ou seja, uma tabela de *hash* de tabelas de *hash*, onde cada ocorrência de autor está presente no primeiro nível, e para cada ocorrência estão todas as ocorrências sem repetidos, com um contador associado. Além da tabela existem uma *string* auxiliar, para guardar os caracteres capturados dos nomes dos autores. Cada nome é inserido num *array* de *strings* sendo depois inseridas todas as associações na tabela de *hash*. No fim, é efetuada um procura na tabela de primeiro nível com o valor do autor escolhido, sendo depois percorrida a tabela de segundo nível desse autor, imprimindo no *stdout* as associações já no formato *Dot*. De igual modo, existem duas variáveis inteiras correspondentes a cada índice de cada *array*. Por último, existe uma *SC* com o nome `AUT` para manipulação do valor do campo autor, nesse contexto.

4.3.1 Expressões Regulares

Antes de se iniciar a descrição, note-se que as *ERs* estão ordenadas de forma a não haver ambiguidade.

4.3.1.1 *START CONDITION* `INITIAL`

- Captura campo autor (*case insensitive*), de forma similar ao que se fez em capítulos anteriores.

  ```
  [Aa][Uu][Tt][Hh][Oo][Rr][ ]*"="[ ]*[{"][ ]*
  ```

  A captura despoleta a inicialização dos índices na posição 0 e inicia *SC* `AUT`.

- Captura tudo o resto, incluindo o caractere *newline*.

  ```
  (.|\n)
  ```

  A ação é ignorar as capturas da *ER*.

4.3.1.2 *START CONDITION* `AUT`

- Captura fim de campo, a semelhança com *ER's* de capítulos anteriores.

  ```
  [ ]*[}"]
  ```

  O procedimento associado a esta *ER* é colocar o caractere `NULL` na última posição da *string* auxiliar, e copiar o valor para o *array* de *strings* auxiliar. O índice deste último é incrementado, sendo de seguida invocada a função `permute` que tratara da reorganização do *array* de *strings* auxiliar, e da respetiva inserção na tabela de *hash*. No fim, volta à *SC* `INITIAL`.

- Captura o separador *and* rodeado de um ou mais espaços e tabulações.

  ```
  [ /t]+(and)[ /t]+
  ```

  Aqui é colocado o caractere `NULL` na última posição da *string* auxiliar, sendo copiada para a posição j do *array* de *strings* auxiliar. A posição j é total de ocorrências de autores até ao momento. O índice da *string* volta a primeira posição, pronto para a leitura de um novo nome.

- Captura tudo o resto, incluindo o caractere *newline*.

  ```
  (.|\n)
  ```

  Aqui é copiada a captura para a *string* auxiliar, incrementado a sua posição.

4.3.2 Estruturas de dados

A estrutura de dados, como já foi mencionado atrás é uma tabela de *hash* multi-nível. A estrutura de base é a mesma que foi utilizada no primeiro capítulo — *uthash*.

4.3.3 Algoritmos

O código que se segue é o utilizado na inserção dos autores na tabela de *hash*.

```
void insertAuthors ( int length )
{int i;
 add_autor(authors[0]);
 for ( i=1; i < length; i++ )
     add_coautor(authors[0], authors[i] );

}
```

O código seguinte é a versão da função `swap` utilizada na reorganização do *array* de *strings*.

```
void swap(char **ap_str1, char **ap_str2)
{char *temp = *ap_str1;
*ap_str1 = *ap_str2;
*ap_str2 = temp;
}
```

A função `permute`, caso haja apenas um autor, insere apenas o mesmo. Se houver mais para cada autor calcula-se uma permutação, tal que, o nome do autor para ser inserido no primeiro nível é colocado à cabeça do *array*, trocando de lugar com o autor seguinte da iteração. Assim obtém-se a permutação pretendida.

```
void permute ( int length )
{
    int j, i;
    if(length==1)
        {
            insertAuthors ( 1 );
```

```
            return;

        }


    for ( i = 0; i < length ; i++ )
        {
            swap( &authors[0], &authors[i] );
            insertAuthors ( length );
        }

    return;
}
```

4.4 Testes e Resultados

4.4.1 Resultados

Utilizando o nome de autor 'Tim Teitelbaum', os resultados estão no apêndice E, na pág. 46. O ficheiro BibTeX utilizado está no apêndice B, pág 38.

A imagem foi obtida através do comando `dot -Tpng res_dot.dot -o out.png`

A escolha do preâmbulo para o *Dot*:

- `strict directed graph`

  Evitar múliplas arestas para o mesmo objeto.

- `ratio=fill` e `size x, y`.

  Dado não caber na página é uniformemente reduzido, e escala para otimizar a razão y/x.

- `ranfir=LR` para colocar o autor do lado esquerdo.

4.4.2 Alternativas, Decisões e Problemas de Implementação

Numa primeira abordagem, tentou-se implementar uma solução que utilizava o `strstr` do `libc`. A solução funcionava, no entanto, não era eficiente. Era necessário ler a linha três vezes: uma para verificar se o autor existia, outra para processar a linha, caso o autor existisse, e o `strstr`. Optou-se pela solução implementada, dado que percorre o documento uma vez, podendo demorar na função `permute` se houver muitos autores. Uma nota final: para poder utilizar esta ferramenta com eficiência, recomenda-se que se normalize primeiro o ficheiro com a ferramenta da parte 2, uma vez que o mesmo autor pode ter o seu nome escrito de diferentes formas.

Conclusão

Neste documento apresentaram-se as propostas de solução para quatro filtros: um contador de entradas bibliográficas, um filtro para normalizar um documento, formatando os nomes dos autores em *N. Apelido* e remoção de chavetas, como também uma ferramenta de *pretty-printing* e o gerador de grafos em *Dot* para coautores de um determinado autor e respetiva densidade de publicação.

Em relação à aprendizagem do Linux, usaram-se diversas ferramentas para obtenção de dados e editores de texto, tal como `grep`, `sort`, *vim*, `cat`, `tac`, `sed`, entre outros. Um ponto comum em todos eles foram as *ER* que ampliaram a visão sobre como trabalhar em Linux.

De igual modo, a utilização das especificações do *Flex* permitiu perceber melhor as *ERs* e a revisita da linguagem C ajudou a limar algumas arestas.

A utilização das estruturas poderia ter sido melhor. A escolha da *uthash* para utilização neste projeto é adequada à sua dimensão, no entanto, existem estruturas mais eficientes para determinados filtros criados.

De um modo geral, os resultados foram animadores, uma vez que se conseguiu cumprir com os requisitos do trabalho, de uma forma eficiente. Muito provavelmente existem melhorias a ser feitas ou algo está parcialmente correta, ou não o está de todo, no entanto, não se tem conhecimento, dados os testes efetuados de situações em que os filtros falharam.

Futuramente, poder-se-á estender o projeto a caracteres especiais e tratamento de caracteres com escape. Uma análise rigorosa poderá por em evidência alternativas de solução mais eficientes ou mais simples. Testes com outras estruturas poderão ser efetuados bem como *benchmarking* de cada estrutura nova aplicada.

Bibliografia

[1] T. H. Cormen. *Introduction to Algorithms*. The MIT Press, 2n edition edition, 2009. ISBN 0262533057.

[2] N. Drakos and R. Moore. Bibtex Entry Types, Field Types and Usage Hints, 1993. URL http://www.openoffice.org/bibliographic/bibtex-defs.pdf.

[3] K. Eleftherios and S. C. Nort. Editing graphs with dotty, 1996. URL http://www.graphviz.org/pdf/dottyguide.pdf.

[4] A. Feder. BibTeX, 2006. URL http://www.bibtex.org/.

[5] P. Lehman, P. Kime, A. Boruvka, and J. Wright. The BibLaTeX Package, 2016. URL http://ftp.eq.uc.pt/software/TeX/macros/latex/contrib/biblatex/doc/biblatex.pdf.

[6] M. E. Lesk and E. Schmidt. Lex: a lexical analyzer generator. 1975. URL http://dinosaur.compilertools.net/lex/index.html.

[7] J. Levine. *Flex & Bison*. 2009. ISBN 978-0-596-15597-1.

[8] J. R. Levine, T. Mason, D. Brown, and T. Niemann. *Lex And Yacc*. O'Reilly, 1992. ISBN 1-56592-000-7.

[9] T. Niemann. Lex & Yacc Tutorial. URL http://epaperpress.com/lexandyacc/.

[10] H. Refsnes, S. Refsnes, K. Jim Refsnes, and J. Egil Refsnes. *Learn HTML and CSS with W3Schools*. Wiley Publishing, Inc., Indianapolis, Indiana, 2010. ISBN 0470611952.

ANEXOS

Apêndice A

Código do *HTML* da Parte 1

```
<!DOCTYPE html>
<html>
<head>
   <style>
     table, th, td {
      border: 1px solid black;
      border-collapse: collapse;
    }
   </style>
</head>
<body>
   <table width=400>
       <tr>
           <td>Article</td>
           <td> 142 </td>
       </tr>
       <tr>
           <td>Booklet</td>
           <td> 0 </td>
       </tr>
       <tr>
           <td>Book</td>
           <td> 47 </td>
       </tr>
       <tr>
           <td>Conference</td>
           <td> 0 </td>
       </tr>
       <tr>
           <td>Inbook</td>
           <td> 3 </td>
       </tr>
       <tr>
           <td>Incollection</td>
           <td> 6 </td>
       </tr>
```

```
37          <tr>
38              <td>Inproceedings</td>
39              <td> 209 </td>
40          </tr>
41          <tr>
42              <td>Manual</td>
43              <td> 13 </td>
44          </tr>
45          <tr>
46            <td>Master's Thesis</td>
47              <td> 2 </td>
48          </tr>
49          <tr>
50              <td>Misc </td>
51              <td> 61 </td>
52          </tr>
53          <tr>
54              <td>Phd Thesis</td>
55              <td> 21 </td>
56          </tr>
57          <tr>
58              <td>Proceedings</td>
59              <td> 4 </td>
60          </tr>
61          <tr>
62              <td>Techreport</td>
63              <td> 140 </td>
64          </tr>
65          <tr>
66              <td>Unpublished</td>
67              <td> 15  </td>
68          </tr>
69  <tr>
70        <td> proceeding </td>
71        <td> 1 </td>
72  </tr>
73     </table>
74  </body>
75  </html>
```

Listing 1: Resultado do *output* da aplicação do filtro na Parte 1

Apêndice B

Código dos ficheiros do BIBTEX para testes

```
@techreport{BW82a,

    year = 1983,
    month = Feb,
    institution = "Institut fur Informatik, TUM",
    note = "(draft version)",
    annote = "espec algebrica",

    author = "Manfred Broy and Martin Wirsing",
    title = "Generalized Heterogeneous Algebras and Partial Interpretations"
    }

@inproceedings{729684,

 title = {Program Slicing Using Weakest Preconditions},
 booktitle = {FME '96: Proceedings of the Third International Symposium of
  ↪  Formal Method
s Europe on Industrial Benefit and Advances in Formal Methods},
 year = {1996},
 isbn = {3-540-60973-3},
 pages = {557--575},
 publisher = {Springer-Verlag},
 address = {London, UK},
 author = {Comuzzi,, Joseph J. and Hart,, Johnson M.},
 }


@inproceedings{1251341,
 author = {David Larochelle and David Evans},
 title = {Statically detecting likely buffer overflow vulnerabilities},
 booktitle = {SSYM'01: Proceedings of the 10th conference on USENIX Security
  ↪  Symposium},
 year = {2001},
 pages = {14--14},
 location = {Washington, D.C.},
```

```
 publisher = {USENIX Association},
 address = {Berkeley, CA, USA}
 }

@book{Guy90a,
    author = "Guy L. Steele",
    title = "Common {L}isp -- The {L}anguage",
    edition = "Second edition",
    year = 1990,
    publisher = dp,
    annote = "prog funcional, linguangens"
    }
@book{ASU86a,
    author = "A. V. Aho and R. Sethi and J. D. Ullman",
    title = "Compilers Principles, Techniques and Tools",
    year = 1986,
    publisher = aw,
    annote = "compilacao"
    }

@phdthesis{Pug88a,
        author = "William W. Pugh",
    title = "Incremental Computation and Incremental Evaluation
                        of Function Programs",
    year = 1988,
    school = "Cornell Univ., Dep. of Computer Science",
    annote = "compilacao incremental, atributos, ambientes prog"
    }

@techreport{Rep92a,
        author = "Thomas Reps",
        title = "Scan Grammars: Parallel Attribute Evaluation
                 Via Data-Parallelism",
        year = 1992,
        month = Nov,
        type = "Research Report",
        institution = "University of Wisconsin - Madison",
        }

@article{RTD83a,
    author = "Thomas Reps and Tim Teitelbaum and A. Demers",
    title = "Incremental Context-Dependent Analysis for Language-based
             Editors",
    journal = "ACM Trans. Programming Languages and Systems (TOPLAS)",
    pages = "449--477",
    year = 1983,
    volume = 5,
    number = 3,
```

```
82      annote = "compilacao incremental, atributos,  ambientes prog"
83      }
84
85  @techreport{KP87a,
86      author = "Kai Koskimies and Jukka Paakki",
87      title = "TOOLS: An Unifying Approach to Object-Oriented
88                Language Interpretation",
89      type = "Research Report",
90      year = 1987,
91      institution = "Univ. of Helsinki, Depart. of Computer Science",
92      note = "(draft)",
93      annote = "compilacao, geradores, atributos, oobjectos"
94      }
95
96  @techreport{CD87a,
97      author = "B. Courcelle and P. Deransart",
98      title = "Proofs for Partial Correctness for Attribute Grammars
99               with Applications to Recursive Procedures and Logic
100              Programming",
101     year = 1987,
102     month = Jan,
103     institution = "Univ. de Bordeaux I, U.E.R. de Mathematiques et
104             Informatiques",
105     number = "I-8702",
106     annote = "compilacao, atributos"
107     }
108
109 @techreport{CFZ80a,
110     author = "B. Courcelle and P. Franchi-Zannettacci",
111     title = "Attribute Grammars and Recursive Program Schemes",
112     type = "rapport de recherche",
113     year = 1980,
114     month = Apr,
115     institution = "Univ. de Bordeaux I",
116     number = "8008",
117     annote = "compilacao, atributos"
118     }
119
120 @ARTICLE{Gram09,
121   AUTHOR =       {GrammaTech},
122   TITLE =        {Dependence Graphs and Program Slicing},
123   YEAR =         {2009}
124 }
```

Listing 2: Ficheiro fonte BibTeX para testes — parte 2 e 3

```
1   @article{RMT86a,
2       author = "Thomas Reps and Carla Marceau and Tim Teitelbaum ",
3       title = "Remote Attribute Updating for Language-based Editors",
```

```
4       journal = "Communications of the ACM",
5       year = 1986,
6       month = Sep,
7       publisher = acm,
8       annote = "compilacao incremental, atributos,  ambientes prog"
9       }
10  @book{RT89b,
11      author = "Thomas Reps and Tim Teitelbaum",
12      title = "The Synthesizer Generator Reference Manual",
13      series = "Texts and Monographs in Computer Science",
14      year = 1989,
15      publisher = sv,
16      annote = "compilacao incremental, atributos,  ambientes prog"
17      }
18  @book{RT89b,
19      author = "Thomas Reps and Tim Teitelbaum",
20      title = "The Synthesizer Generator Reference Manual",
21      series = "Texts and Monographs in Computer Science",
22      year = 1989,
23      publisher = sv,
24      annote = "compilacao incremental, atributos,  ambientes prog"
25      }
26  @article{RTD83a,
27      author = "Thomas Reps and Tim Teitelbaum and A. Demers",
28      title = "Incremental Context-Dependent Analysis for Language-based
29               Editors",
30      journal = "ACM Trans. Programming Languages and Systems (TOPLAS)",
31      pages = "449--477",
32      year = 1983,
33      volume = 5,
34      number = 3,
35      publisher = acm,
36      annote = "compilacao incremental, atributos,  ambientes prog"
37      }
38
39  @article{TR81a,
40      author = "Tim Teitelbaum and Thomas Reps",
41      title = "The Cornell Program Synthesizer: A Syntax-Directed
42               Programming Environment",
43      journal = "Communications of the ACM",
44      year = 1981,
45      month = Sep,
46      volume = 24,
47      number = 9,
48      publisher = acm,
49      annote = "compilacao incremental, atributos,  ambientes prog"
50      }
51  @techreport{Kos89b,
```

```
52      author = "Kai Koskimies",
53      title = "Techniques for Modular Language Implementation",
54      type = "Research Report",
55      year = 1989,
56      institution = "Univ. of Tampere, Depart. of Computer Science",
57      number = "A-1989-5",
58      annote = "compilacao, sintaxe, parsing, modular"
59      }
60
61  @techreport{Kos89a,
62      author = "Kai Koskimies",
63      title = "Lazy Recursive Descent Parsing for Modular
64                Language Implementation",
65      type = "Research Report",
66      year = 1989,
67      institution = "Gesellschaft fur Mathematik und Datenverarbeitung mbH",
68      number = "Arbeitspapiere der GMD 376",
69      annote = "compilacao, sintaxe, parsing, modular"
70      }
```

Listing 3: Ficheiro fonte BIBTEX para testes — parte 4

Apêndice C

Resultado da Parte 2

```
@techreport{BW82a,

    year = 1983,
    month = Feb,
    institution = {Institut fur Informatik, TUM},
    note = {(draft version)},
    annote = {espec algebrica},

    author = {M. Broy and M. Wirsing},
    title = {Generalized Heterogeneous Algebras and Partial Interpretations}
    }

@inproceedings{729684,

 title = {Program Slicing Using Weakest Preconditions},
 booktitle = {FME '96: Proceedings of the Third International Symposium of
  ↪ Formal Method
s Europe on Industrial Benefit and Advances in Formal Methods},
 year = {1996},
 isbn = {3-540-60973-3},
 pages = {557--575},
 publisher = {Springer-Verlag},
 address = {London, UK},
 author = {J. J. Comuzzi and J. M. Hart},
 }


@inproceedings{1251341,
 author = {D. Larochelle and D. Evans},
 title = {Statically detecting likely buffer overflow vulnerabilities},
 booktitle = {SSYM'01: Proceedings of the 10th conference on USENIX Security
  ↪ Symposium},
 year = {2001},
 pages = {14--14},
 location = {Washington, D.C.},
 publisher = {USENIX Association},
```

```
 address = {Berkeley, CA, USA}
 }

@book{Guy90a,
    author = {G. L. Steele},
    title = {Common {L}isp -- The {L}anguage},
    edition = {Second edition},
    year = 1990,
    publisher = dp,
    annote = {prog funcional, linguangens}
    }
@book{ASU86a,
    author = {A. V. Aho and R. Sethi and J. D. Ullman},
    title = {Compilers Principles, Techniques and Tools},
    year = 1986,
    publisher = aw,
    annote = {compilacao}
    }

@phdthesis{Pug88a,
        author = {W. W. Pugh},
    title = {Incremental Computation and Incremental Evaluation
                        of Function Programs},
    year = 1988,
    school = {Cornell Univ., Dep. of Computer Science},
    annote = {compilacao incremental, atributos, ambientes prog}
    }

@techreport{Rep92a,
        author = {T. Reps},
        title = {Scan Grammars: Parallel Attribute Evaluation
                 Via Data-Parallelism},
        year = 1992,
        month = Nov,
        type = {Research Report},
        institution = {University of Wisconsin - Madison},
        }

@article{RTD83a,
    author = {T. Reps and T. Teitelbaum and A. Demers},
    title = {Incremental Context-Dependent Analysis for Language-based
             Editors},
    journal = {ACM Trans. Programming Languages and Systems (TOPLAS)},
    pages = {449--477},
    year = 1983,
    volume = 5,
    number = 3,
    annote = {compilacao incremental, atributos,  ambientes prog}
```

```
83      }
84
85  @techreport{KP87a,
86      author = {K. Koskimies and J. Paakki},
87      title = {TOOLS: An Unifying Approach to Object-Oriented
88                Language Interpretation},
89      type = {Research Report},
90      year = 1987,
91      institution = {Univ. of Helsinki, Depart. of Computer Science},
92      note = {(draft)},
93      annote = {compilacao, geradores, atributos, oobjectos}
94      }
95
96  @techreport{CD87a,
97      author = {B. Courcelle and P. Deransart},
98      title = {Proofs for Partial Correctness for Attribute Grammars
99               with Applications to Recursive Procedures and Logic
100              Programming},
101     year = 1987,
102     month = Jan,
103     institution = {Univ. de Bordeaux I, U.E.R. de Mathematiques et
104            Informatiques},
105     number = {I-8702},
106     annote = {compilacao, atributos}
107     }
108
109 @techreport{CFZ80a,
110     author = {B. Courcelle and P. Franchi-Zannettacci},
111     title = {Attribute Grammars and Recursive Program Schemes},
112     type = {rapport de recherche},
113     year = 1980,
114     month = Apr,
115     institution = {Univ. de Bordeaux I},
116     number = {8008},
117     annote = {compilacao, atributos}
118     }
119
120 @ARTICLE{Gram09,
121   AUTHOR =       {GrammaTech},
122   TITLE =        {Dependence Graphs and Program Slicing},
123   YEAR =         {2009}
124 }
```

Listing 4: Resultado do *output* da aplicação do filtro na Parte 2

Apêndice D

Resultado da Parte 3

```
1  #################
2  Autor(es) -----:          Manfred Broy
3                            Martin Wirsing
4  Titulo --------:          Generalized Heterogeneous Algebras and Partial
     ↪  Interpretations
5  Ano ---------- :          1983
6  Mes ---------- :          Feb
7  Instituicao ---:          Institut fur Informatik
8
9
10
11 #################
12 Autor(es) -----:          Comuzzi,, Joseph J.
13                           Hart,, Johnson M.
14 Titulo --------:          Program Slicing Using Weakest Preconditions
15 Titulo do livro:          FME '96: Proceedings of the Third International
     ↪  Symposium of Formal Methods Europe on Industrial Benefit and Advances
     ↪  in Formal Methods
16 Ano ---------- :          1996
17 Pags ----------:          557--575
18 Publicação de -:          Springer-Verlag
19 Morada --------:          London
20
21
22
23 #################
24 Autor(es) -----:          David Larochelle
25                           David Evans
26 Titulo --------:          Statically detecting likely buffer overflow
     ↪  vulnerabilities
27 Titulo do livro:          SSYM'01: Proceedings of the 10th conference on
     ↪  USENIX Security Symposium
28 Ano ---------- :          2001
29 Pags ----------:          14--14
30 Publicação de -:          USENIX Association
31 Morada --------:          Berkeley
```

```
#################
Autor(es) -----:         Guy L. Steele
Titulo --------:         Common Lisp -- The Language
Edicao --------:         Second edition
Ano ---------- :         1990
Publicação de -:         dp


#################
Autor(es) -----:         A. V. Aho
                         R. Sethi
                         J. D. Ullman
Titulo --------:         Compilers Principles, Techniques and Tools
Ano ---------- :         1986
Publicação de -:         aw


#################
Autor(es) -----:         William W. Pugh
Titulo --------:         Incremental Computation and Incremental Evaluation
  ↪  of Function Programs
Ano ---------- :         1988
Escola --------:         Cornell Univ.


#################
Autor(es) -----:         Thomas Reps
Titulo --------:         Scan Grammars: Parallel Attribute Evaluation Via
  ↪  Data-Parallelism
Ano ---------- :         1992
Mes ---------- :         Nov
Tipo ----------:         Research Report
Instituicao ---:         University of Wisconsin - Madison


#################
Autor(es) -----:         Thomas Reps
                         Tim Teitelbaum
                         A. Demers
Titulo --------:         Incremental Context-Dependent Analysis for
  ↪  Language-based Editors
```

```
Jornal --------:         ACM Trans. Programming Languages and Systems
  ↪  (TOPLAS)
Pags ----------:         449--477
Ano ---------- :         1983
Volume --------:         5
Numero --------:         3


#################
Autor(es) -----:         Kai Koskimies
                         Jukka Paakki
Titulo --------:         TOOLS: An Unifying Approach to Object-Oriented
  ↪  Language Interpretation
Tipo ----------:         Research Report
Ano ---------- :         1987
Instituicao ---:         Univ. of Helsinki


#################
Autor(es) -----:         B. Courcelle
                         P. Deransart
Titulo --------:         Proofs for Partial Correctness for Attribute
  ↪  Grammars with Applications to Recursive Procedures and Logic
  ↪  Programming
Ano ---------- :         1987
Mes ---------- :         Jan
Instituicao ---:         Univ. de Bordeaux I
Numero --------:         I-8702


#################
Autor(es) -----:         B. Courcelle
                         P. Franchi-Zannettacci
Titulo --------:         Attribute Grammars and Recursive Program Schemes
Tipo ----------:         rapport de recherche
Ano ---------- :         1980
Mes ---------- :         Apr
Instituicao ---:         Univ. de Bordeaux I
Numero --------:         8008


#################
Autor(es) -----:         GrammaTech
Titulo --------:         Dependence Graphs and Program Slicing
```

```
121 Ano ---------- :          2009
```

Listing 5: Resultado do *output* da aplicação do filtro na Parte 3

Apêndice E

Resultado da Parte 4

```
1 strict digraph G {
2   ratio=fill;
3   size="7.5,10"
4   center=true;
5   rankdir=LR;
6 "Tim Teitelbaum" -> "Thomas Reps"                [label = 5        ];
7 "Tim Teitelbaum" -> "Carla Marceau"                [label = 1        ];
8 "Tim Teitelbaum" -> "A. Demers"                [label = 1        ];
9 }
```

Listing 6: Resultado do *output* da aplicação do filtro na Parte 3

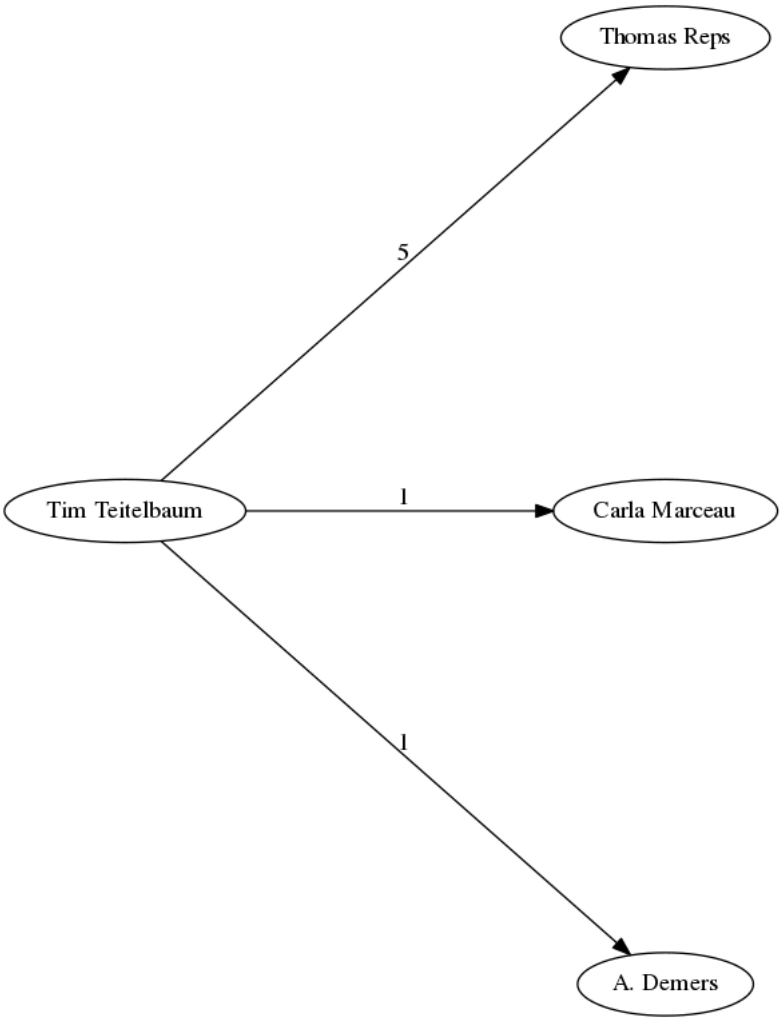

Figura E.1: Grafo Resultante da aplicação do filtro da Parte 4