Processamento de Linguagens (3º ano de MIEI)

Trabalho Prático 3

Relatório de Desenvolvimento

Catarina Machado (a81047)

Gonçalo Faria (a86264)

João Vilaça (a82339)

10 de Junho de 2019

Resumo

O terceiro e final trabalho prático no âmbito da unidade curricular Processamento de Linguagens consistiu na elaboração de um reconhecedor de Thesaurus ISO 2788. Adicionalmente o programa desenvolvido transforma a especificação de uma ontologia em Thesaurus ISO 2788 numa representação interna e representa-a graficamente com html.

No presente relatório são apresentadas as decisões conceptuais e lógicas tomadas na construção do reconhecedor, gramática tradutora e estrutura interna.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Área: Processamento de Linguagens

1.1 Enquadramento e Contexto

A eficácia de um índice de conceitos como forma de identificar e recuperar documentos depende de uma adequadamente construída e usada linguagem de indexação. Para representar hierarquias de relacionamentos a priori entre conceitos, genericamente denominadas ontologias, constantes num dado índice foi concebida uma linguagem para este efeito designada Thesaurus ISO 2788.

Na referida ontologia é composta por conceitos, propriedades e relacionamentos entre conceitos. Adicionalmente, é possível nesta estabelecer predicados de implicações entre relações, que possibilitam a inferência de adicionais relacionamentos entre conceitos.

Neste projeto, pretende-se numa primeira fase desenvolver um reconhecedor de Thesaurus ISO 2788, de seguida construir uma representação interna da ontologia inerente e por fim, através desta, desenvolver uma página html que a represente.

1.2 Problema

Escrever uma gramática independente de contexto para o reconhecimento de ontologias no formato T2788 e o respectivo analisador léxico. Criar um programa que processe uma especificação de uma ontologia e construa uma representação interna através de Yacc, com recurso a uma gramática tradutora. Por fim, com a representação interna criada efectuar uma travessia e gerar uma página HTML para cada conceito, sendo construídas hiperligações de acordo com as relações conceptuais.

1.3 Objectivo

O objectivo do presente relatório é o de documentar as decisões de desenvolvimento da solução de software apresentada e demonstrar esta em funcionamento.

1.4 Estrutura do Relatório

O presente relatório está dividido em 5 diferentes capítulos.

No capítulo 1, **Introdução**, é feito um enquadramento e contextualização do trabalho prático e, em seguida, é feita uma descrição do problema a desenvolver, assim como os objetivos do projeto e do relatório.

No capítulo 2, **Análise e Especificação**, é feita uma descrição informal do problema e uma especificação dos requisitos necessários para uma correta resolução do problema, ou seja, é desvendada a abordagem que consideramos ser a mais correta para uma eficiente resolução e explicação do trabalho prático.

Em seguida, no capítulo 3, **Concepção e desenho da Resolução** é feita uma descrição detalhada de todo o desenvolvimento do projeto até se obter a solução final.

Posteriormente, no capítulo 4, **Testes**, são apresentados alguns *prints* de todas as páginas HTML geradas e dos grafos obtidos para um determinado exemplo de input.

Por fim, no capítulo 5, **Conclusão**, termina-se o relatório com uma síntese e análise do que trabalho realizado.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

Desenvolver um reconhecedor de Thesaurus ISO 2788. Com recurso a uma gramática tradutora desenvolver uma página web que represente graficamente uma ontologia especificada.

2.2 Especificação do Requisitos

- Analisador sintáctico
- Gramática independente de contexto
- Estrutura de dados da representação interna
- Gramática tradutora

Capítulo 3

Concepção/desenho da Resolução

3.1 Estruturas de Dados

A estrutura interna criada, responsável por representar cada um das ontologias dadas, foi um grafo. Um grafo é genericamente constituído por um conjunto de vértices e um conjunto de arcos. Os arcos são um subconjunto do produto cartesiano dos vértices, ou seja uma relação entre os vértices. Os vértices representam os conceitos e os arcos os relacionamentos.

Em 3.1 encontra-se uma representação simbólica da estrutura de dados Ontologia. Os registos, correspondentes a cada uma das componentes da cadeia de produtos cartesianos, são $NomeConceito$ que é uma cadeia de caracteres com o nome da ontologia, $Map < NomeConceito, Conceito >$ que contém o dicionário de conceitos usados em toda a ontologia, $Set < Linguagem >$ onde se encontram todas as linguagens suportadas pela ontologia, $Set < Propriedades >$ o conjunto de propriedades constantes na ontologia e $Map < Relacão, Set < Relacão >>$ que contém as correspondências entre as relações constantes na ontologia e o conjunto de relações que cada uma delas implica.

```
Ontologia = {
    NomeOntologia,
    Map<NomeConceito,Conceito>,
    Set<Linguagem>,
    Set<Propriedade>,
    Map<Relação,Set<Relação>>
}
```

Em 3.1 encontra-se uma representação simbólica da estrutura de dados *Conceito*. Os seus registos são $NomeConceito$ que é uma cadeia de caracteres com o nome do conceito em questão, $Map < Relacão, Set < Conceito >>$ que corresponde ao conjunto de adjacência, organizado por relação em que a cada relação faz corresponder um conjunto de conceitos, $Map < Linguagem, NomeConceitoTraduzido >$ que é um dicionário que a cada Linguagem, na forma de uma cadeia de caracteres, corresponde a uma tradução do conceito nessa linguagem e $Map < Property, Descricão >$ que é um dicionário que a cada propriedade corresponde uma descrição, sendo que tanto a propriedade como a descrição são uma cadeia de caracteres.

```
Conceito = {
    NomeConceito,
    Map<Relação,Set<Conceito>>,
    Map<Linguagem, NomeConceitoTraduzido>,
```

```
        Map<Property, Descrição>
    }
```

Por fim a *Relacão* foi apenas representada por uma cadeia de caracteres com o respectivo identificador desta.

3.2 Implementação

A construção do reconhecedor sintático de Thesaurus ISO 2788, foi iniciada com a construção de um analisador léxico, de seguida uma gramática independente de contexto, que contêm a lógica associada à linguagem que se pretende reconhecer, e por fim, através de uma gramática tradutora, é gerada uma representação interna da ontologia e esta é usada para criar uma representação gráfica em html.

3.2.1 Analisador Sintático

O analisador léxico desempenhou a função de traduzir a sequência de caracteres de entrada num conjunto de símbolos léxicos pré determinados que constituem as componentes da linguagem a reconhecer. Os símbolos que identificados na linguagem desenvolvida foram $LANGDEC$, $BASELANGDEC$, $RELATE$, $TITLE$, $DESCRIPTOR$, $SEPARATOR$, $LANG$, $WORD$, $NEWLINE$. Associado a cada um desses símbolo existe uma expressão regular que encapsula todos as instanciações da respetiva componente.

```
\%language                  { return LANGDEC;}
\%baselang                  { return BASELANGDEC;}
\%inv                       { return RELATE; }
\%title                     { return TITLE; }
\%descriptor                { return DESCRIPTOR;}
\#[^\n]*                    { ; }
,                           { return SEPARATOR;}
[A-Z][A-Z]+/[" "\n#]        {
                                yylval.string = strdup(yytext);
                                return LANG;
                            }
[a-zA-Z()+/\-&$!?]+         {
                                yylval.string = strdup(yytext);
                                return WORD;
                            }
\%                          { BEGIN NONDEFINED; }
<NONDEFINED>[^\n]+/\n       { BEGIN INITIAL; }
\n\n+                       { return NEWLINE; }
" "|\n  { ; }
```

3.2.2 Gramática Independente de Contexto

A abordagem tomada para a criação da gramática independente de contexto que melhor representa linguagem a reconhecer foi uma *top-down*.

Cada especificação de uma ontologia encontra-se essencialmente dividida em duas partes, uma em que são descritas propriedades gerais da ontologia(*Specs*) e outra em que se encontram descritos os conceitos(*Concepts*).

```
Thesaurus   : Specs Concepts
            ;
```

A parte onde se encontram os *meta-dados*, as propriedades gerais da ontologia, é posteriormente composta por uma sequência de especificações atómicas. Cada uma das especificações atómicas ora descreve o conjuntos de linguagens suportadas, a linguagem base, o título, implicações de relações ou o conjunto de identificadores de propriedades da ontologia.

```
Specs       : Specs Spec
            | &
            ;
Spec        : LANGDEC Langs
            | BASELANGDEC LANG
            | RELATE LANG LANG
            | TITLE Terms
            | DESCRIPTOR Langs
            ;
```

A outra parte, a que compreende a descrição e enumeração dos conceitos da ontologia, é subdividida numa sequências de conceitos. Cada conceito é iniciado com uma quebra de linha e uma palavra, essa palavra corresponderá ao identificador do respectivo conceito. Adicionalmente, após a palavra poderá constar um conjunto de campos, aqui designados por *Properties*.

```
Concepts    : Concepts Concept
            | Concept
            ;
Concept     : NEWLINE WORD Properties
            | NEWLINE WORD
            ;
```

Os campos são essencialmente uma sequência de símbolo não terminal *Property*. Cada *Property* ora é uma tradução, uma descrição ou uma relação a um conjuto de conceitos. Como tanto o identificador de linguagem, secção de descrição e relação são representados pelo mesmo símbolo léxico($LANG$) esta regra de produção é bastante genérica.

```
Properties  : Properties Property
            | Property
            ;
Property    : LANG ConceptList
            ;
ConceptList : ConceptList SEPARATOR Terms
            | Terms
            ;
```

Adicionalmente foram usados outros símbolos não terminais para auxiliar a simplificar a gramática resultante.

```
Langs       : Langs LANG
            | LANG
            ;

Terms       : Terms WORD
            | WORD
            ;
```

3.2.3 Gramática Tradutora

Através de **yacc** foi criada uma gramática tradutora que tem como objectivo instanciar a estrutura de dados representativa da ontologia contida num dado ficheiro de entrada.

A gramática usada foi uma extensão da apresentada na secção 3.2.2 devido à necessidade de diferenciar os diferentes tipos de campos de cada conceito.

O símbolo não terminal inicial, o *Thesaurus*, foi expandido para que no início da execução a variável global *saurus*, a variável que instanciará a representação interna da ontologia, fosse inicializada. A acção associada a este símbolo é a da criação da hierarquia de ficheiros html, que compreendem a página que representa graficamente a ontologia, e a susequente libertação da memória pela variável *saurus*.

```
Thesaurus   : Start Specs Concepts Remaining    {
                showOntology(saurus);
                unmkOntology(saurus);
            }
            ;
Start       : &                                 {
                saurus = mkOntology();
            }
            ;
Remaining   : Remaining NEWLINE                 { ; }
            | NEWLINE                           { ; }
            ;
```

Em essência, as ações implementadas apenas chamaram o método do módulo C que implementa a incorporação das diferentes características da ontologia na estrutura de dados que a representa.

```
Spec        : LANGDEC Langs                     { ; }
            | BASELANGDEC LANG                  {
                setBaseLanguage(saurus,$2);
            }
            | RELATE LANG LANG                  {
                relateRaw(saurus,$2,$3);
            }
            | TITLE Terms                       {
                setTitle(saurus,$2);
            }
            | DESCRIPTOR NSections              { ; }
            ;

Langs       : Langs LANG                        {
                addLanguage(saurus, $2);
            }
            | LANG                              {
                addLanguage(saurus, $1);
            }
            ;
```

```
NSections   : NSections LANG                    {
                addNoteSection(saurus, $2);
            }
            | LANG                              {
                addNoteSection(saurus, $1);
            }
            ;
```

Por fim, símbolos não terminais como *Conceito* que necessitam de informação contida em vários símbolos não terminais contêm acções ligeiramente mais sofisticadas. Neste caso, sendo que o símbolo *Properties* contém uma lista de tuplos, em que uma das componentes é o nome de uma relação e a outra é uma lista de nomes de conceitos, esta função percorre cada um dos tuplos e associa conceitos por meio da relação descriminada.

```
Concept     : NEWLINE WORD Properties           {
                Concept cpt = getConcept(saurus, $2);

                for(GList* cur = $3; cur; cur = cur->next){
                    Prop tup = (Prop)cur->data;

                    for(GList* innercur = tup->concepts; innercur; innercur= innercur->next )
                        associate(saurus, cpt, tup->r, (char *)innercur->data);

                    unmkProp(tup);
                }

                g_list_free($3);
            }
            | NEWLINE WORD                      {
                getConcept(saurus, $2);
            }
            ;
```

3.2.4 Módulo Ontology

A estrutura de dados que representa cada ontologia foi implementada na linguagem C. A sua API encontra-se descriminada em baixo.

```
typedef struct ontology *Ontology;

Ontology mkOntology();
void unmkOntology( Ontology saurus );

void setBaseLanguage( Ontology saurus, const char * lang);
void addLanguage( Ontology saurus, const char * lang);
void setTitle( Ontology saurus, const char* title);
void addNoteSection( Ontology saurus, const char* notesection);

Relation getRelation( Ontology saurus, const char* relationname );
Concept getConcept( Ontology saurus, const char* conceptname );

void relateRaw( Ontology saurus, const char* subclasse, const char* superclasse);
void relate( Ontology saurus, Relation subclasse, Relation superclasse);
void associate( Ontology saurus, Concept source, const char * relation, const char * assoc);

void showOntology( Ontology saurus);
```

3.2.5 Compilação e Execução

Depois de todo o código terminado, elaboramos uma Makefile que compila o código efetuado, com a opção **clean** e com a opção **dots**. A opção **dots** executa a Makefile gerada pela execução do programa, que permitirá obter os ficheiros **png** através de **dot**.

Para facilitar o processo de compilar e executar o programa recorremos ao seguinte shell script, intitulado **run**:

```
#!/bin/bash
make
./thesaurus < $1
make dots
open out/html/index.html
```

Neste script começa-se por compilar o código, corrê-lo com o ficheiro que pretendemos analisar, gerar os png e, por fim, abrir a página inicial do programa.

Desta forma, basta correr o seguinte comando no terminal, que corre o script apresentado anteriormente, onde o argumento corresponde ao ficheiro que contém a ontologia.

```
./run exemplo.th
```

Não esquecer que inicialmente será necessário atribuir as permissões necessárias ao ficheiro **run**, que pode ser feito através do comando seguinte.

```
> chmod 755 run
```

Capítulo 4

Testes

Para testar o código efetuado elaboramos um exemplo de uma ontologia, apresentada em seguida:

```
%language PT EN FR # linguas: PT EN
%baselang EN # lingua de base: EN
%inv NT BT # NT, BT sao relacoes inversas a NT B => b BT a
%inv BT NT
%descriptor SOUND SN
%title Guantanamo Bay

#conceitos:
animal # termo na baselang
PT animal # LINGUA termo
FR animale
NT cat, dog, cow, # NT = narrow term = termo especifico
fish, ant, human # new
NT camel # new
BT Life being # BT = broader term = termo generico
# linha em branco : separador de conceitos

cat
PT gato
SN animal que tem sete vidas e m(e)ia
SOUND miau miau
RT lion

#comentario # desde o simbolo '#' ate ao fim da linha
```

Em seguida iremos apresentar o resultado que é obtido após a execução do nosso programa, utilizando esta ontologia como argumento.

4.1 Resultados

4.1.1 Impressão da Página Inicial

A página inicial do programa é a seguinte:

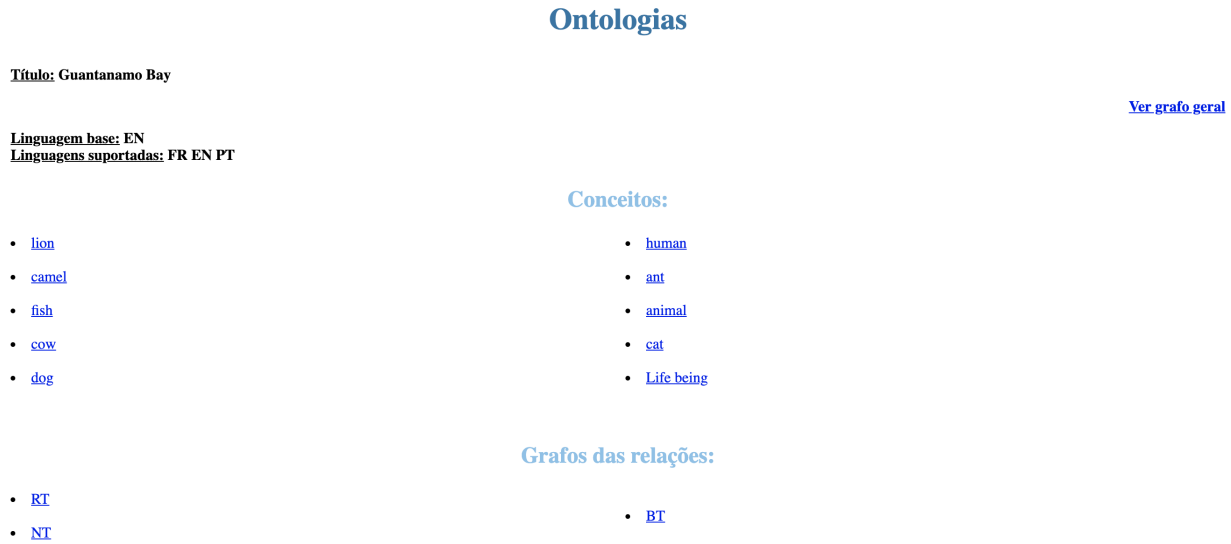

Figura 4.1: Página inicial

Através dessa página é possível visualizar o título da ontologia, a linguagem base, as linguagens suportadas, visualizar o grafo geral das relações entre os conceitos, selecionar um dos conceitos para assim poder obter mais informações acerca do mesmo e ainda visualizar o grafo de uma determinada relação.

O output html da página apresentada anteriormente é o seguinte:

```
<!DOCTYPE html>
<html>
<head>
        <title>PL</title>
        <meta name='viewport' content='width=device-width, initial-scale=1'>
        <style>
                body { max-width: 1300px; margin: auto; }
                .colunas { line-height: 200margin-left: 60px; -webkit-column-count: 2;
                           -moz-column-count: 2; column-count: 2; }
        </style>
</head>
<body>
        <h1><p align='center'><font color='#2874A6'>Ontologias</font></p></h1>
        <h4>
                <u>Título:</u> Guantanamo Bay</br>
                <p align='right'><a href="grafo.html">Ver grafo geral</a></p>
                <u>Linguagem base:</u> EN</br>
                <u>Linguagens suportadas:</u> FR EN PT
        </h4>
        <h2><p align='center'><font color='#85C1E9'>Conceitos:</font></p></h2>
        <div class='colunas'>
                <li><a href="lion.html">lion</a></li></br>
                <li><a href="camel.html">camel</a></li></br>
```

```
                <li><a href="fish.html">fish</a></li></br>
                <li><a href="cow.html">cow</a></li></br>
                <li><a href="dog.html">dog</a></li></br>
                <li><a href="human.html">human</a></li></br>
                <li><a href="ant.html">ant</a></li></br>
                <li><a href="animal.html">animal</a></li></br>
                <li><a href="cat.html">cat</a></li></br>
                <li><a href="Life-being.html">Life being</a></li></br>
        </div>
        <h2><p align='center'><font color='#85C1E9'>Grafos das relações:</font></p></h2>
        <div class='colunas'>
                <li><a href="RTgrafo.html">RT</a></li></br>
                <li><a href="NTgrafo.html">NT</a></li></br>
                <li><a href="BTgrafo.html">BT</a></li></br>
        </div>
</body>
</html>
```

4.1.2 Grafo geral

Através da Figura 4.1, se clicarmos em "Ver grafo geral", o utilizador é redireccionado para uma nova página onde é possível visualizar o grafo geral da ontologia, que contém todos os conceitos e todas as relações existentes entre eles. O grafo apresentado encontra-se na Figura 4.2.

4.1.3 Conceito

Tal como se pode ver através da Figura 4.1, existe na página inicial uma lista com todos os conceitos da ontologia. Clicando num desses conceitos, por exemplo no conceito **cat**, é possível visualizar todas as informações do mesmo.

A página obtida é a que se apresenta na Figura 4.3.

4.1.4 Grafos das relações

É ainda possível visualizar o grafo de uma determinada relação entre conceitos. Tal como se pode ver na Figura 4.1, é possível escolher qual a relação que pretendemos visualizar.

Escolhendo, por exemplo, a relação **BT**, a página obtida é a que se encontra na Figura 4.4.

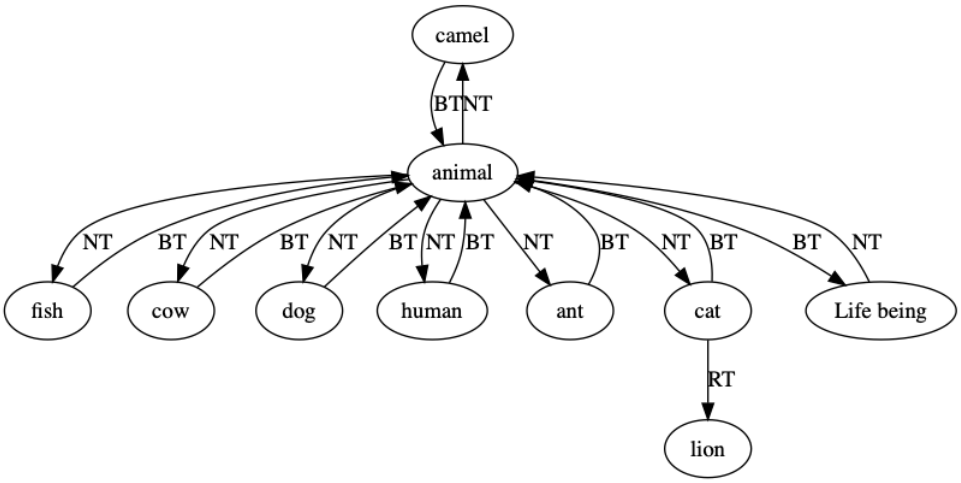

Figura 4.2: Grafo geral

cat

Traduções:

- PT - gato

Notas:

- SOUND - miau miau
- SN - animal que tem sete vidas e m(e)ia

Relações:

RT:

- lion

BT:

- animal

Figura 4.3: Conceito: cat

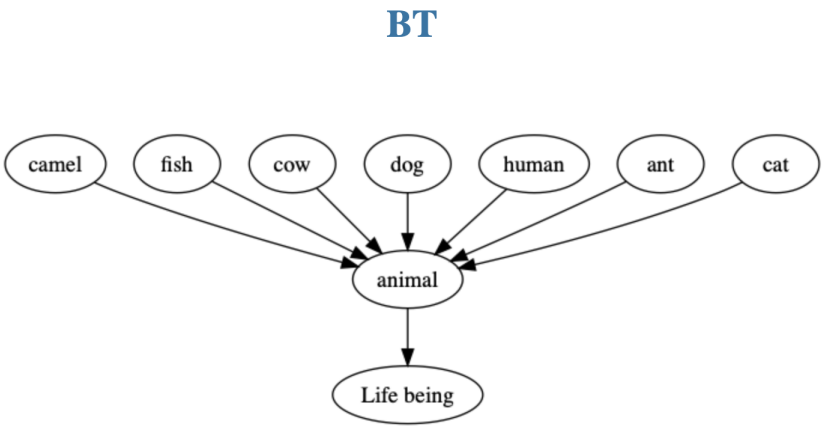

Figura 4.4: Grafo da relação BT

Capítulo 5

Conclusão

A solução de software ultimada permite, de uma forma simples e eficaz, reconhecer uma especificação em Thesaurus ISO 2788. Adicionalmente, sendo que a especificação descreve uma hierarquia complexa de relacionamentos entre conceitos, com recurso a html, através de uma gramática tradutora, é criada uma página web. A página web, permite a consulta de conceitos e também a visualização de um conjunto de diagramas, sendo cada um a representação de uma das hierarquias de relacionamentos.