U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Inteligência Artificial
EIC0029

Ano Letivo 2015/2016

Relatório Final

“Bibliogenie”

Grupo D5_2

Diogo Belarmino Coelho Marques
up201305642@fe.up.pt

Pedro Miguel Pereira de Melo
up201305618@fe.up.pt

29 de Maio de 2016

Índice

1 Objetivo

Este projeto foi desenvolvido no âmbito da unidade curricular de Inteligência Artificial (*IART*) inserida no 3º ano do *Mestrado Integrado em Engenharia Informática e Computação* (*MIEIC*) e consistiu na implementação de um programa (ao qual foi dado o nome de *Bibliogenie*) que permite obter várias informações relativas a escritores e suas respetivas obras literárias.

O *Bibliogenie* é capaz de interpretar diversos tipos de questões colocadas pelo utilizador em linguagem natural e de procurar informação numa base de conhecimento interna ao programa os dados relevantes à resolução das mesmas por forma a devolver ao utilizador uma resposta adequada.

2 Especificação

Nesta secção será descrito com detalhe a abordagem tomada para a resolução dos problemas e para a construção da aplicação.

2.1 Possíveis cenários de utilização

O *Bibliogenie* suporta dois tipos de frases: interrogativas e declarativas.

Abaixo foram colocadas duas listas com exemplos de frases válidas e de ambos os tipos. Estes exemplos tentam abordar o maior número possível de casos de utilização distintos, não representando a totalidade de frases aceites.

Nota: devido aos problemas com caracteres acentuados descritos posteriormente neste documento, algumas das palavras contidas nos exemplos seguintes não são válidas, dado serem acentuadas, pelo que têm de ser modificadas para poderem ser utilizadas.

2.1.1 Exemplos de frases interrogativas

- ➔ Que escritor português escreveu “A Morgadinha dos Canaviais”?
- ➔ Quem escreveu “O Eterno Marido”?
- ➔ Que romances e comédias foram escritos por Camilo Castelo Branco?
- ➔ Quando foi escrito “O Adolescente”?
- ➔ Que livros foram publicados em inglês?
- ➔ Quantos livros foram escritos em português?
- ➔ “O Conde de Monte Cristo” foi escrito em 1854?
- ➔ Que escritores viveram no século XX?
- ➔ Quais os autores que viveram depois do século XVIII?
- ➔ Que escritores estão vivos?
- ➔ Quando nasceu Fernando Pessoa?
- ➔ Quem nasceu em 1917?
- ➔ Quais os escritores que nasceram antes do século XX?

2.1.2 Exemplos de frases declarativas

- ➔ Camilo Castelo Branco escreveu romances.
- ➔ Jane Austen é inglesa.
- ➔ Fernando Pessoa é europeu.
- ➔ A “Dama das Camélias” foi escrita em francês.
- ➔ “Romeu e Julieta” foi publicado no século XVI.
- ➔ “A Morgadinha dos Canaviais” é um romance.
- ➔ Oscar Wilde nasceu em 1864.
- ➔ Emily Brontë nasceu depois de 1800.
- ➔ Luís de Camões nasceu antes de Mia Couto.

- ➔ William Shakespeare viveu antes do século 'XVIII'.
- ➔ Eric Blair faleceu em 1950.
- ➔ Arthur Clarke faleceu antes de Virgílio Ferreira.

2.2 Abordagem ao problema

Às frases do tipo declarativo apenas deve ser dada uma resposta do tipo *sim/não* ou *verdadeiro/falso*.

Existem dois tipos de respostas possíveis para as frases do tipo interrogativo: qualitativas e quantitativas. Uma resposta do primeiro tipo consiste numa lista de um ou mais elementos, contendo informações sobre autores, obras ou datas. Uma resposta do tipo quantitativo consiste apenas num valor numérico.

Ambas os tipos de frase podem ou não conter adjetivos e outros complementos linguísticos além dos substantivos e verbos.

O *Bibliogenie* suporta a inclusão de múltiplos comandos, desde que estes se encontrem separados pela partícula "*e*". Para tal torna-se forçosamente necessário guardar os contextos das várias frases. Usaram-se assim as *DCG*s (do inglês *Definite Clause Grammar*), dado que estas permitem a expressão de linguagens naturais em linguagens de programação tais como *Prolog*. O *Bibliogenie* deve ainda verificar a sintaxe e a semântica das frases inseridas, aceitando e respondendo somente aquelas que estejam em concordância com a gramática definida.

A base de conhecimento inerente ao *Bibliogenie* deve estar implementada por forma a ser de fácil pesquisa e robusta. Torna-se assim necessário garantir que um conjunto de dados não é descrito mais do que uma vez e que a informação aí disponível não é ambígua.

A aplicação é constituida por duas componentes: uma *interface* gráfica em linguagem *Java* e um servidor em *Prolog* que contêm a base de conhecimento.

Os dados contidos nessa base de conhecimento encontram-se divididos em dois grandes grupos, consoante sejam relativos a um autor ou a uma obra literária. Dessa forma, de cada escritor é possível saber-se o primeiro e último nome, o ano de nascimento, o ano da sua morte no caso de ainda não ter falecido, o seu sexo, a sua nacionalidade e finalmente, os seus pseudónimos, se algum. De cada obra literária é possível saber-se o título, o autor, o ano de publicação, o género literário em que se insere e o título da coleção a que pertence, se alguma.

A interação com os utilizadores é assegurada pela *interface* gráfica descrita. A ligação com o servidor é realizada através de um servidor *HTTP* que se encontrará a correr na nossa aplicação, e a *interface* gráfica funcionará como cliente, estabelecendo uma ligação com este servidor e enviando pedidos ao mesmo.

Entre as várias funcionalidades da *interface* gráfica, destaca-se a possibilidade de guardar o histórico de interrogações da sessão atual num ficheiro de texto e de carregar um ficheiro de texto com várias questões que depois podem ser executadas sequencialmente.

A linguagem em que as frases devem ser expressas é o português, que contêm palavras acentuadas, não aceites pelo *Prolog*. Dessa forma, para garantir o funcionamento da aplicação, removeram-se todos os acentos das palavras.

3 Desenvolvimento

3.1 Ferramentas / *API*s utilizadas

Durante o desenvolvimento do *Bibliogenie* utilizou-se o *IDE NetBeans* para a criação da *interface* gráfica da aplicação na linguagem de programação *Java*, tendo-se recorrido para tal às bibliotecas *Swing* e *Abstract Window Toolkit*. A comunicação entre a aplicação do cliente e *Prolog* está a ser assegurada pela biblioteca *Jasper* incluída no *SICStus* que serve de *interface* com aplicações Java.

Em paralelo utilizou-se SICStus Prolog para a construção da gramática de cláusulas definidas, na representação da base de conhecimento e na criação das interrogações de acesso à mesma.

3.2 Estrutura e módulos da aplicação

O código da aplicação encontra-se dividido em duas grandes partes; uma relativa ao cliente em *Java* e a outra relativa ao servidor *Prolog*. Em seguida serão discriminados as classes e os módulos de cada uma dessas partes.

3.2.1 Estrutura do cliente

O código da aplicação cliente encontra-se dividido nos seguintes pacotes:

- **Application:** contém a lógica da interface gráfica do cliente e de processamento de eventos do utilizador;
- **Model:** contém a parte do código responsável por manter em Java uma imagem da base de conhecimento implementada em *Prolog*;
- **Resources:** contém as imagens utilizadas na aplicação.

3.2.2 Estrutura do servidor

O servidor da aplicação contém o código escrito na linguagem *Prolog* e encontra-se dividido nos seguintes módulos:

- **Bibliogenie.pl**: contém a base de conhecimento da aplicação, encontrando-se aqui declarada toda a informação relativa aos diversos autores e seus pseudónimos assim como das suas obras e coleções;
- **Gramatica.pl**: contém a especificação das regras gramaticais que as frases devem seguir para serem válidas e consequentemente respondidas;
- **Lexico.pl**: contém a declaração de todas as palavras aceites pela aplicação;

- **Query.pl:** contém todo o código utilizado na pesquisa e obtenção de informação da base de conhecimento.

3.3 Detalhes da implementação

3.3.1 Detalhes da implementação do servidor

Seguem-se os detalhes relativos à implementação dos módulos do servidor.

3.3.1.1 Bibliogenie.pl

Para representar informação relativa a autores utilizou-se o predicado ***autor/8:***

autor(dost, 'Fiodor', 'Dostoievski', 1821, 1881, m,rus, []).
autor(pess, 'Fernando', 'Pessoa', 1888, 1935, m,por, [soar]).

- O primeiro argumento corresponde ao **identificador (único) do autor**, que permite não só distinguir esse autor de entre os restantes como também impedir que dois autores com o mesmo nome estejam na origem de conflitos;
- O segundo argumento corresponde ao **primeiro nome** do autor;
- O terceiro argumento corresponde ao **último nome** do autor;
- O quarto argumento corresponde ao **ano de nascimento** do autor;
- O quinto argumento corresponde ao **ano de morte** do autor. Se um autor ainda for vivo este argumento deve tomar o valor -1;
- O sexto argumento corresponde ao **sexo do autor**. Optou-se pelo uso de "*m*" para os homens e de "*f*" para as mulheres;
- O sétimo argumento corresponde ao **identificador único da nacionalidade** do autor, que permitirá conhecer o país de nascimento de um autor;
- O oitavo argumento consiste numa lista de **identificadores (únicos) dos pseudónimos** desse autor, caso existam. Na ausência de pseudónimos para um determinado autor a lista de estar vazia.

Para cada identificador único da lista de pseudónimos acima descrita deve existir uma chamada ao predicado ***pseudo/3***:

pseudo(soar, 'Bernardo', 'Soares').
pseudo(gowr, 'George', 'Orwell').

Para armazenar a informação relativa às obras de um autor criou-se o predicado ***livro*/6**:

livro(dost-cri, 'Crime e Castigo', dost, 1866, romance, null).
livro(cacb-mis1, 'Mistérios de Lisboa - I', cacb, 1854, romance, cacb-misl).

- ➢ O primeiro argumento corresponde ao **identificador** (**único) da obra,** permitindo não só distingui-la das restantes obras mas também impedir que duas obras literárias com o mesmo nome estejam na origem de qualquer ambiguidade;
- ➢ O segundo argumento corresponde ao **título** da obra;
- ➢ O terceiro argumento corresponde ao **identificador (único) do autor**;
- ➢ O quarto argumento corresponde ao **ano de publicação** da obra;
- ➢ O quinto argumento corresponde ao **género literário** da obra;
- ➢ O sexto argumento, consiste num **identificador (único) da colecção** ou série em que esta obra se insere. Quando não aplicável deve ser colocado a *null*.

Para cada identificar único da lista de coleções acima referido, deve existir uma chamada ao predicado **coleccao/3**.

coleccao(cacb-lnpd, 'Livro Negro de Padre Dinis', cacb).

Admite-se assim que o livro com identificador *cacb-lnpd,* escrito pelo autor de identificar *cacb*, pertence à coleção de nome *"Livro Negro de Padre Dinis"*.

Criaram-se ainda os predicado **pais/5** e **pais_continente/2** para facilitar o acesso à informação de cariz geográfica de um autor.

O primeiro predicado permite obter informação sobre um dado país.

pais(jpn,'Japao',japones, 'japonês', 'japonesa').
pais(moz,'Moçambique', portugues, 'moçambicano', 'moçambicana').
pais(por,'Portugal', portugues, 'português', 'portuguesa').

- ➢ O primeiro argumento corresponde ao **identificador (único) da nacionalidade** do autor;
- ➢ O segundo argumento corresponde ao **nome do país**;
- ➢ O terceiro argumento corresponde ao **nome da língua** desse país;
- ➢ O quarto argumento corresponde ao **nome *singular* dado aos cidadãos do sexo masculino** desse país;
- ➢ O quinto argumento corresponde ao **nome *singular* dado aos cidadãos do sexo feminino** desse país.

O segundo predicado permite associar o um dado país ao seu continente.

pais_continente(jpn, asia).
pais_continente(moz, africa).
pais_continente(por, europa).

- ➢ O primeiro argumento corresponde ao **identificador (único) da nacionalidade** do autor;

➢ O segundo argumento corresponde ao **nome do continente** onde o país se encontra.

3.3.1.2 Gramatica.pl

Para permitir o uso da partículo “e” no interior de uma frase criaram-se os predicados **contx_cont/2**, **contx_nac/2** e **contx_gen/2**, que permitem listar continentes, nacionalidades e géneros literários. São eles que permitem a validação de frases como “Quais os escritores portugueses **e** ingleses **e** franceses que...”, “Quais os escritores europeus **e** americanos que...” ou “Que comédias **e** dramas **e** romances escreveu... ”. Os predicados foram implementados de uma forma quase idêntica, que pode ser generalizada da seguinte forma:

contx_XXXX(N-G,[H|T]) --> [e], subst_XXXX(N-G, H), (contx_cont(N-G, T) ; []).
contx_cont(_-_,[]) --> [].

Torna-se assim facilmente observável que o predicado é recursivo e que os sucessivos valores vão sendo guardados numa lista [H|T].

Para o processamento de todos os substantivos - qualificados ou não - criou-se o predicado **nom/6**:

nom(N-G,Adj,Nom, Nac, Cont, _) --> subst(N-G,Nom), atributo(N-G, Adj, Nac, Cont).
nom(N-G,Adj,_, Nac, Cont, Gen) --> subst_gen(Gen,N-G), atributo(N-G, Adj, Nac, Cont).
nom(N-G,_,Nom,_,_,_) --> subst(N-G,Nom).
nom(N-G,_,_,_,_,[Gen|T]) --> subst_gen(Gen,N-G), (contx_gen(N-G,T) ; []).

➢ O primeiro argumento corresponde ao **grupo número-género** do substantivo;
➢ O segundo argumento corresponde ao **significado** do adjetivo, caso existente;
➢ O terceiro argumento corresponde ao **significado** do substantivo. Diferentes substantivos podem ter o mesmo significado
➢ O quarto argumento corresponde à lista de **nacionalidades** associadas ao substantivo;
➢ O quinto argumento corresponde à lista de **continentes** associados ao substantivo;
➢ O sexto argumento corresponde à lista de **géneros literários** associados ao substantivo.

Para o processamento de complementos determinativos criou-se o predicado **com_deter/6**, implementado da seguinte forma:

com_deter(N-G, Adj, Nom, Nac, Cont, Gen) → preposicao(N-G), nom(N-G, Adj, Nom, Nac, Cont, Gen).

Através deste predicado, partículas frásicas como “... dos autores portugues...” ou “... do livro mais antigo ...” conseguem ser validadas.

3.3.1.3 Lexico.pl

O predicado **subst/3**, exemplificado em baixo, permite definir quais os substantivos a ser aceites pelo *Bibliogenie***:**

subst(s-m, autor) --> [autor].
subst(s-f, autor) --> [autora].
subst(p-m, autor) --> [autores].
subst(p-f, autor) --> [autoras].

subst(s-m, autor) --> [escritor]**.**
subst(s-f, autor) --> [escritora].
subst(p-m, autor) --> [escritores].
subst(p-f, autor) --> [escritoras].

- O primeiro argumento corresponde ao **grupo número-género** do substantivo, que conduz à necessidade de se invocar o predicado várias vezes para o mesmo substantivo;
- O segundo argumento corresponde ao **significado** do substantivo. Diferentes substantivos podem ter o mesmo significado.

Para além de *autor* e *escritor*, a aplicação suporta ainda os seguintes substantivos (listados no singular): *coleção, saga, trilogia, volume, criptonimo, heterónimo, ortonimo, morte, falecimento, nascimento, ano, século, idioma, língua, nacionalidade* e *idade*.

O predicado **subst_gen/3**, permite definir quais os géneros literários aceites pela aplicação**:**

subst_gen(livro,s-m) --> [livro].
subst_gen(livro,p-m) --> [livros].

- O primeiro argumento corresponde ao **género literário**. Os géneros "*livro*" e "*obra*" são genéricos, sendo utilizados quando se pretende referir a todos os possíveis géneros;
- O segundo argumento corresponde ao **grupo número-género,** que conduz à necessidade de se invocar o predicado várias vezes para o mesmo género literário;

A aplicação suporta os seguintes géneros literários (listados no singular): *comédia, crónica, conto, drama, ficção científica, manga, poesia, prosa, romance, teatro* e *tragédia*.

As nacionalidades aceites são declaradas através do predicado **subst_nac/3:**

subst_nac(chi, s-m) --> [chileno].
subst_nac(chi, s-f) --> [chilena].
subst_nac(chi, p-m) --> [chilenos].

subst_nac(chi, p-f) --> [chilenas].

- ➢ O primeiro argumento corresponde ao **código do país**;
- ➢ O segundo argumento corresponde ao **grupo número-género,** que conduz à necessidade de se invocar o predicado várias vezes para a mesma nacionalidade.

A aplicação suporta as seguintes nacionalidades: *alemã, chilena, colombiana, espanhola, francesa, inglesa, indiana, irlandesa, japonesa, moçambicana, portuguesa, russa* e *americana*.

O *Bibligenie* só suporta três tempos verbais, o *presente*, o *pretérito perfeito* e o *pretérito imperfeito*, o que apenas através do predicado **verbo/3**.

verbo(s, escrever, passado) --> [escreveu].
verbo(p, escrever, passado) --> [escreveram].

verbo(s, escrever, passado) --> [publicou].
verbo(p, escrever, passado) --> [publicaram].

verbo(s, escrever, passado) --> [redigiu].
verbo(p, escrever, passado) --> [redigiram].

- ➢ O primeiro argumento corresponde ao **número**;
- ➢ O segundo argumento corresponde ao **significado geral do verbo**. Diferentes verbos podem ter o mesmo significado;
- ➢ O terceiro argumento visa indicar se a acção ocorreu no **presente ou no passado**.

Apenas são suportados os seguintes verbos: *ser, viver, publicar, escrever, redigir, morrer, falecer* e *nascer*.

Todos estes verbos podem ser ainda apresentados na forma passiva, o que é definido pelo predicado **verbo_passiva/2**:

verbo_passiva(s-m, escrever) --> [redigido].
verbo_passiva(s-f, escrever) --> [redigida].
verbo_passiva(p-m, escrever) --> [redigidos].
verbo_passiva(p-f, escrever) --> [redigidas].

- ➢ O primeiro argumento corresponde ao **grupo número-género**;
- ➢ O segundo argumento corresponde ao **significado geral do verbo**.

O uso de adjetivos é assegurado pelo predicado **adjetivo/2**:

adjetivo(s-_, recente) --> [recente].
adjetivo(p-_, recente) --> [recentes].

adjetivo(s-m, antigo) --> [antigo].
adjetivo(s-f, antigo) --> [antiga].
adjetivo(p-m, antigo) --> [antigos].
adjetivo(p-f, antigo) --> [antigas].

- O primeiro argumento corresponde ao **grupo número-género**;
- O segundo argumento corresponde ao **significado geral do adjetivo**.

A aplicação suporta, de momento, apenas os adjetivos *recente* e *antigo*.

O uso dos graus superlativo de inferioridade e superlativo de superioridade destes adjetivos também é possível através do predicado **adj_comp/3**:

adj_comp(mais, recente, maisrecente).
adj_comp(menos, recente, menosrecente).
adj_comp(mais,antigo, maisantigo).
adj_comp(menos,antigo, menosantigo).

- O primeiro argumento corresponde ao indica qual **o grau superlativo** a ser usado;
- O segundo argumento corresponde ao **adjetivo**;
- O terceiro argumento corresponde a uma código utilizado na gramática durante o processamento de adjetivos no grau especificado.

3.3.2.4 Query.pl

Os predicados apresentados nas tabelas seguintes permitem obter informações relativas aos autores:

PREDICADOS(s)	FUNÇÃO
autores_sem_pseudonimos(Lista) **autores_n_pseudonimos**(N, Lista)	obtém a lista dos autores sem pseudónimos ou com exactamente N pseudónimos;
autores_nascidos_seculo(=, Seculo, Lista) **autores_nascidos_seculo**(>, Seculo, Lista) **autores_nascidos_seculo**(<, Seculo, Lista)	obtém a lista dos autores que nasceram durante, antes ou depois de um dado século;
autores_mortos_seculo(=, Seculo, Lista) **autores_mortos_seculo**(>, Seculo, Lista) **autores_mortos_seculo**(<, Seculo, Lista)	obtém a lista dos autores que morreram durante, antes ou depois de um dado século;
autores_vivos_seculo(=, Seculo, Lista) **autores_vivos_seculo**(>, Seculo, Lista) **autores_vivos_seculo**(<, Seculo, Lista)	obtém a lista dos autores que viveram durante, antes ou depois de um dado século;
autores_nacionalidade(Nacionalidade, Lista)	obtém a lista dos autores que

autores_idioma(Idioma, Lista)	nasceram num dado país;
comparar_autores_novo(Autor1, Autor2, Autor)	verifica qual de dois autores é o mais velho;
autores_mesmo_seculo(AnoNascimento1, AnoNascimento2) **autores_mesma_idade**(Autor1, Autor2)	verifica se dois autores nasceram no mesmo século ou ano;
autores_seculos_diferentes(Autor1, Autor2) **autores_anos_diferentes**(Autor1, Autor2)	verifica se dois autores nasceram em séculos ou anos diferentes;
autores_mesmo_continente(Autor1, Autor2) **autores_mesma_nacionalidade**(Autor1, Autor2)	verifica se dois autores são do mesmo continente ou nacionalidade;
autores_continentes_diferentes(Autor1,Autor2) **autores_nacionalidades_diferentes**(Autor1, Autor2)	verifica se dois autores são de continentes ou nacionalidades diferentes;
verificar_pseudonimo(Pseudonimo, Autor)	verifica se um dado pseudónimo é de determinado autor;
autor_mais_velho(Autor) **autor_mais_novo**(Autor)	indica qual o autor mais velho/novo na base de conhecimento;

Os predicados na tabela seguinte permitem obter informações relativas aos livros:

PREDICADO(s)	**FUNÇÃO**
livros_mesmo_ano(Livro1, Livro2) **livros_mesmo_seculo**(Livro1, Livro2)	verifica os se dois livros Livro1, Livro2 foram publicados no mesmo ano ou século;
livros_anos_diferentes(Livro1, Livro2) **livros_seculos_diferentes**(Livro1, Livro2)	verifica se os dois livros Livro1, Livro2 foram publicados em anos ou séculos diferentes;
livros_genero(Genero, Lista)	obtém a lista com todos os livros de um dado género literário;
livros_publicados_entre(LimiteInferior, LimiteSuperior, Lista)	obtém a lista com todos os livros publicados entre dois anos por um dado;
livros_publicados_antes(Autor, AnoLimite, Lista) **livros_publicados_ano**(Autor, AnoLimite, Lista) **livros_publicados_entre**(Autor,	obtém a lista com todos os livros de um dado autor que foram publicados antes/depois de um certo ano ou que foram publicados entre dois anos dados;

LimiteInferior, LimiteSuperior, Lista)	
livro_mais_recente(Livro) **livro_mais_antigo**(Livro)	indica qual o livros mais recente/antigo presente na base de conhecimento;
livros_coleccao(Coleccao, Lista) **livros_publicados**(Autor, Lista)	obtém a lista de todos os livros de uma dada coleção ou de um determinado autor;
livros_publicados(Lista)	obtém a lista com todos os livros presentes na base de conhecimento;

4 Conclusões

Analisando os resultados obtidos nas experiências efetuadas aquando da implementação é possível concluir que o *Bibliogenie* consegue analisar vários tipos de frase e em diversos contextos, conseguindo compreender qual é a informação querida pelo utilizador, procurar por essa informação na base de conhecimento para depois a apresentar.

5 Melhoramentos

O principal melhoramento a ser executado sobre a aplicação é o de aumentar o vocabulário por ela aceite, permitindo ao utilizador colocar questões numa linguagem ainda mais semelhante com aquela que por ele é utilizada no seu dia a dia.

Em particular, destaca-se o possível reconhecimento da partícula “não”, o que permitiria a colocação de frases como “Que romances *não* foram escritos por Camilo Castelo Branco?” ou “ ‘A Morgadinha dos Canaviais’ *não* foi escrita em russo.”.

Destaca-se ainda a possível aceitação de valores numéricos ordinais, o que tornaria possível a colocação de questões como “Qual foi o *segundo* livro escrito por Jane Austen após 1780?”

6 Recursos

6.1 Divisão do trabalho

Ambos os membros do grupo participaram quer na implementação do *Bibliogenie* quer na elaboração deste documento, nas proporções abaixo referidas:

- *Diogo Belarmino Marques:* **60 %**
- *Pedro Miguel Melo:* **40 %**

6.2 Webgrafia

- Apontamentos das aulas teóricas sobre representação de conhecimento e linguagem natural, disponibilizados pelo Professor Eugénio Oliveira no sítio oficial da unidade curricular (https://paginas.fe.up.pt/~eol/IA/1516/ia_.html);
- **Swedish Institute of Computer Science**. 1998. “SICStus Prolog - Built-In Predicates” https://sicstus.sics.se/sicstus/docs/3.7.1/html/sicstus_10.html;
- **Swedish Institute of Computer Science**. 2007. “ref-gru - SICStus Prolog” https://sicstus.sics.se/sicstus/docs/4.0.2/html/sicstus/ref_002dgru.html#ref_002dgru;
- **ISO 3166-1 - Wikipédia**. Acedido em 19/05/2016 às 14:13. https://en.wikipedia.org/wiki/ISO_3166-1