Trabalho Prático 1
Relatório de Desenvolvimento
SKOS

Miguel Pereira Fernandes (44024)

9 de Abril de 2014

Conteúdo

Capítulo 1

Introdução

1.1 Enunciado

O Simple Knowledge Organization System (SKOS) é um vocabulário RDF (Resource Des- cription Framework) para representar especificações de conhecimento semi-formais, tais como thesauri, taxonomias, sistemas de classificação ou listas finitas de termos, às vezes designadas vocabulários controlados. Como o SKOS é baseado em RDF, as especificações SKOS podem ser lidas e interpretadas por máquinas e podem ser trocadas entre aplicações de software. O SKOS foi concebido com o objetivo de facilitar a migração de modelos organizacionais exis- tentes para a Web Semântica. No entanto, pode ser usado para especificar novos modelos de conhecimento e partilhá-los na Web. Pode ser usado isoladamente ou combinado com outras linguagens mais formais como o OWL (Ontology Web Language). Pode também ser usado como ponte entre as linguagens de ontologias como o OWL e as pouco estruturadas ferramentas que suportam a Web social. Para uma documentação mais abrangente os alunos deverão consultar os documentos oficiais do W3C: http://www.w3.org/TR/skos-primer/ Neste projeto, pretende-se que seja desenvolvido um processador que, recebendo um ficheiro SKOS contendo uma especificação de um modelo de conhecimento, crie um conjunto de páginas HTML que permitam uma navegação fácil no modelo. Listam-se a seguir duas ontologias SKOS que os alunos deverão usar como casos de estudo: Ontologia das localidades portuguesas : http://www.di.uminho.pt/ jcr/XML/didac/ xmldocs/SKOS/localidades.rdf; Ontologia informática da ACM : http://www.di.uminho.pt/ jcr/XML/-didac/xmldocs/ SKOS/ACM-SKOSTaxonomy.xml.

1.2 Objetivos

Este trabalho prático tem como principais objectivos:

- aumentar a experiência de uso do ambiente Linux, da linguagem imperativa C (para codi- ficação das estruturas de dados e respectivos algoritmos de manipulação), e de algumas ferramentas de apoio à programação;
- aumentar a capacidade de escrever Expressões Regulares (ER) para descrição de padrões de frases;
- desenvolver, a partir de ERs, sistemática e automaticamente Processadores de Linguagens Regulares, que filtrem ou transformem textos;
- utilizar geradores de filtros/processadores de texto, como o Flex
- compreensão do modelo de dados SKOS e desenvolvimento de processador de texto que converta esse modelo num conjunto de páginas web navegáveis.

1.3 Contextualização

O formato SKOS apesar da sua imensa potencialidade na área de troca de informação é um formato pouco inteligível e de difícil navegação ao utilizador comum. Torna-se por isso necessário desenvolver ferramentas que, partindo do modelo de dados SKOS, consigam extrair a sua informação vital e a consigam converter para formatos mais amigáveis. Um dos formatos que melhor permite visualizar essa informação é o html. O html é largamente utilizado hoje em dia. Quer seja para consultar o email, ler notícias, ver vídeos ou ouvir música, quase toda a gente já acedeu a uma página web pelo menos uma vez na vida. O html permite ainda integrar outras tecnologias como javascript e css que permitem pegar em dados crus e criar interfaces agradáveis e de utilização intuitiva. Uma vez que o modelo de dados SKOS tem uma estrutura bem definida o que se pretende implementar é um processador de texto que receba como entrada um ficheiro SKOS, extraia a informação mais relevante e a insira numa estrutura de dados criada para o efeito. Será depois desenvolvida uma rotina que a partir da estrutura criada gerará um conjunto de páginas navegáveis e um pequeno índice de conceitos.

Capítulo 2

Implementação

2.1 Decisões

Após análise da estrutura dos dois casos de estudo fornecidos e também da estrutura global do modelo de dados SKOS identificaram-se os seguintes elementos como sendo os elementos chave do modelo de dados:

- skos:ConceptScheme - contém informação geral do modelo nomeadamente:
  - dc:title - titulo do modelo
  - dc:date - data do modelo
  - skos:hasTopConcept - são conceitos de alto nível. Servem quase como índice dos restantes conceitos. Estas "etiquetas" possuem um atributo, "rdf:resource", que contém um identificador do "skos.Concept" que contém a informação mais detalhada do skos.TopConcept.
- skos:Concept - se pensarmos num modelo SKOS como numa abstração de um grafo, o skos:Concept corresponde aos nodos desse grafo. Contém um atributo com identificador que representa de forma única o nodo e contém ainda as seguintes sub-etiquetas:
  - skos:prefLabel - contém o titulo do nodo. Existe apenas um nó deste tipo por linguagem. (Para este trabalho desconsiderou-se a linguagem uma vez que ambos os casos de estudo continham apenas a especificação para uma única linguagem)
  - skos:altLabel - contém designações alternativas para o nodo.
  - skos:narrower - contém um atributo que estabelece uma ligação hierárquica com um outro nodo mais restrito do que o nodo a ser especificado.
  - skos:broader - inverso do skos:narrower. Contém um atributo que estabelece uma ligação hierárquica com um nodo mais abrangente do que o nodo a ser especificado.

– skos:related - define uma ligação associativa entre dois nodos. [1]

Uma vez que estes foram considerados os elementos chave de um SKOS, serão estes os elementos processados pelo autómato definido em Flex.

2.2 Estruturas de dados e funções chave

Em seguida serão explicitados os extratos do código considerado mais fulcral ao funcionamento da aplicação. Para consultar ou realizar download do código fonte poderão consultar o seguinte repositório no github `https://github.com/migpfernandes/pl_tp1.git` .

2.2.1 Listas Genéricas

Uma vez que um ficheiro SKOS pode ter um número indefinido de "Concepts" é necessário utilizar uma estrutura dinâmica capaz de armazenar tantos dados quanto o necessário. Como será também necessário utilizar listas de "Strings" para armazenar os vários elementos de relacionamento optou-se por criar uma lista genérica assim como funções genéricas de manuseamento de listas. A estrutura que define um nó da lista é a seguinte:

```
typedef struct node_s {
  void *data;
  struct node_s *next;
} NODE;
```

As funções abaixo são as que habitualmente encontramos na interação com listas: inserção, remoção e pesquisa. As funções que têm um sufixo "_file" recebem como argumento um ficheiro para onde escrevem dados resultantes da execução das funções também passadas como argumentos. As funções que têm um prefixo "_global" recebem para além do nó a processar uma lista completa dos nós para que possam fazer travessias e obter informações adicionais necessárias à execução de funções.

```
NODE *list_create(void *data);
NODE *list_insert_after(NODE *node, void *data);
NODE *list_insert_beginning(NODE *list, void *data);
NODE *list_insert_sorted(NODE *list, void *data,int(*func)(void*,void*))
    ;
int list_remove(NODE *list, NODE *node);
int list_foreach(NODE *node, int(*func)(void*));
int list_foreach_file(NODE *node,FILE * file, int(*func)(void*,FILE*));
int list_foreach_global(NODE *node,NODE* list,char* title, int(*func)(
    void*,void*,char*));
int list_foreach_global_file(NODE *node,NODE* list,FILE* file, int(*func
    )(void*,void*,FILE*));
NODE *list_find(NODE *node, int(*func)(void*,void*), void *data);
```

2.2.2 Concept

A estrutura abaixo basicamente corresponde à estrutura definida no modelo de dados SKOS de um "Concept". Um "Concept" tem um id, uma prefLabel,

pode ter uma ou mais designações alternativas e vários apontadores para os "Concepts" relacionados, sejam estes "broader", "related" ou "narrower".

```
typedef struct concept_s {
    char* Id;
    char* prefLabel;
    NODE altLabel;  //Lista de strings
    NODE *relatedIDs;  //Lista de strings
    NODE *narrowerIDs;  //Lista de strings
    NODE *broaderIDs;  //Lista de strings
} *Concept,ConceptN;
```

A única função diretamente relacionada à estrutura acima é a "initConcept" que tem como objetivo apenas alocar espaço e definir o ID de um novo "Concept". As restantes propriedades são preenchidas utilizando as funções de listas demonstradas anteriormente.

```
Concept initConcept(char *ID);
```

2.2.3 Global

A estrutura "Global" basicamente encapsula as estruturas anteriores serve para armazenar toda a informação considerada útil do ficheiro SKOS.

```
typedef struct sGlobal {
    char* title;
    char* data;
    NODE *topConcepts;  //Lista de strings
    NODE *concepts;  //Lista de concepts
} Global;
```

Após terminado o processamento de texto a variável global conterá todos os dados que se pretendem transpostos para páginas html. Para realizar essa transposição criou-se a rotina "geraPaginas".

```
void geraPaginas(Global global);
NODE* addConcept(NODE* list,Concept conc);
```

2.3 FLEX

2.3.1 Declaração de variáveis globais

```
%{
#include "concepts.h"
#include "list.h"
#include "ctype.h"
Global global;
Concept conc;
%}
```

2.3.2 Contextos utilizados para processamento do texto

```
%x concept conceptScheme topConcept preflabel altlabel
```

O nome dos contextos permite facilmente identificam as entidades que pretendem processar. Existe uma grande proximidade entre os contextos e as etiquetas do ficheiro. Para além de todos estes contextos existe ainda o contexto raiz, identificado por 0, que basicamente descarta todos os carateres até aparecer uma expressão que inicie um dos outros contextos.

2.3.3 Processamento dos nodos

```
<concept>rdf\:about\=\"[^\"]+\"
{
char* id = strdup(yytext);
id[strlen(id)-1] = '\0';
id = id + 11;
removeNonAlfanumericChars(id);
conc = initConcept(id);
}

<concept>{PREFLABELBTAG}
{ BEGIN preflabel; }

<concept>{ALTLABELBTAG}
{ BEGIN altlabel; }

<concept>{CONCEPTETAG}
{ global.concepts = list_insert_sorted(global.concepts,conc,
    conceptComparer); }

<concept>{BROADERTAG}
{
char *id = strdup(yytext);
id[strlen(id)-3]='\0';
id = id + 28;
removeNonAlfanumericChars(id);
conc->broaderIDs= list_insert_beginning(conc->broaderIDs,id);
}
```

Capítulo 3

Casos de estudo

3.1 Ficheiro SKOS a processar

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
<!ENTITY skos "http://www.w3.org/2004/02/skos/core#">
<!ENTITY dc "http://purl.org/dc/elements/1.1/">
]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:skos="http://www.w3.org/2004/02/skos/core#" xmlns:dc="http://
          purl.org/dc/elements/1.1/">
     <skos:ConceptScheme rdf:about="http://totem.semedica.com/taxonomy/
          The ACM Computing Classification System (CCS)">
          <dc:title>The ACM Computing Classification System (CCS)</
               dc:title>
          <dc:date>2012</dc:date>
          <skos:hasTopConcept rdf:resource="#10002944"/>
          <skos:hasTopConcept rdf:resource="#10002950"/>
          <skos:hasTopConcept rdf:resource="#10002951"/>
          <skos:hasTopConcept rdf:resource="#10002978"/>
          <skos:hasTopConcept rdf:resource="#10003033"/>
          <skos:hasTopConcept rdf:resource="#10003120"/>
          <skos:hasTopConcept rdf:resource="#10003456"/>
          <skos:hasTopConcept rdf:resource="#10003752"/>
          <skos:hasTopConcept rdf:resource="#10010147"/>
          <skos:hasTopConcept rdf:resource="#10010405"/>
          <skos:hasTopConcept rdf:resource="#10010520"/>
          <skos:hasTopConcept rdf:resource="#10010583"/>
          <skos:hasTopConcept rdf:resource="#10011007"/>
          <skos:hasTopConcept rdf:resource="#10011641"/>
     </skos:ConceptScheme>
     <skos:Concept rdf:about="#10002944" xml:lang="en">
          <skos:prefLabel xml:lang="en">General and reference</
               skos:prefLabel>
          <skos:altLabel xml:lang="en">general and reference works</
               skos:altLabel>
          <skos:inScheme rdf:resource="http://totem.semedica.com/taxonomy/
               The ACM Computing Classification System (CCS)"/>
          <skos:topConceptOf
               rdf:resource="http://totem.semedica.com/taxonomy/The ACM
                    Computing Classification System (CCS)"/>
```

```
        <skos:narrower rdf:resource="#10011122"/>
        <skos:narrower rdf:resource="#10011123"/>
    </skos:Concept>
    <skos:Concept rdf:about="#10002945" xml:lang="en">
        <skos:prefLabel xml:lang="en">Surveys and overviews</
            skos:prefLabel>
        <skos:altLabel xml:lang="en">surveys</skos:altLabel>
        <skos:altLabel xml:lang="en">overview articles</skos:altLabel>
        <skos:altLabel xml:lang="en">overviews</skos:altLabel>
        <skos:inScheme rdf:resource="http://totem.semedica.com/taxonomy/
            The ACM Computing Classification System (CCS)"/>
        <skos:broader rdf:resource="#10011122"/>
    </skos:Concept>
....
</rdf:RDF>
```

3.2 Processamento do ficheiro

Para executar o programa executa-se a seguinte instrução numa bash:

```
./skosProcessor < ACM-SKOSTaxonomy.xml
```

NOTA: é necessário garantir que existe uma pasta chamada concepts na pasta do programa.

3.3 Resultado do processamento

3.3.1 index.html

```
<html xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns# xmlns:rdfs=
    http://www.w3.org/2000/01/rdf-schema# xmlns:skos=http://www.w3.org
    /2004/02/skos/core# xmlns:dc=http://purl.org/dc/elements/1.1/>
<head>
    <meta http-equiv=Content-Type content=text/html; charset=Western (
        Windows 1252)>
    <title>The ACM Computing Classification System (CCS)</title>
</head>
<body>
  <h1>The ACM Computing Classification System (CCS)</h1>
<table width=100%>
  <tr>
    <td width=30%>
    <h3>Concept Index</h3>
    <ol>
      <li><a href=concepts/10011681.html>3-tier architectures</a></li>
      <li><a href=concepts/10010239.html>3D imaging</a></li>
      <li><a href=concepts/10010601.html>3D integrated circuits</a></li>
      <li><a href=concepts/10011351.html>A. van Wijngaarden</a></li>
      <li><a href=concepts/10011328.html>ABET, Inc.</a></li>
      <li><a href=concepts/10011573.html>ANSI C</a></li>
      <li><a href=concepts/10011161.html>AOL, Inc.</a></li>
      <li><a href=concepts/10011051.html>API languages</a></li>
      <li><a href=concepts/10011164.html>AT&#38;T</a></li>
      <li><a href=concepts/10003202.html>Abstract data types</a></li>
      <li><a href=concepts/10010622.html>Abstract machines</a></li>
      <li><a href=concepts/10011119.html>Abstraction</a></li>
      <li><a href=concepts/10010211.html>Abstraction and micro-operators
          </a></li>
      <li><a href=concepts/10011682.html>Abstraction, modeling and
          modularity</a></li>
      <li><a href=concepts/10011153.html>Accenture</a></li>
```

```
      <li><a href=concepts/10003565.html>Acceptable use policy
          restrictions</a></li>
      <li><a href=concepts/10011109.html>Acceptance testing</a></li>
      <li><a href=concepts/10002993.html>Access control</a></li>
      <li><a href=concepts/10011683.html>Access protection</a></li>
      <li><a href=concepts/10011738.html>Accessibility</a></li>
    </ol>
    </td>
    <td width=70% valign=top>
    <h3>Top Concepts</h3>
<ul>
    <li><a href=concepts/10011641.html>Proper nouns&#58; People,
        technologies and companies</a></li>
    <li><a href=concepts/10011007.html>Software and its engineering</a>
        </li>
    <li><a href=concepts/10010583.html>Hardware</a></li>
    <li><a href=concepts/10010520.html>Computer systems organization</a>
        </li>
    <li><a href=concepts/10010405.html>Applied computing</a></li>
    <li><a href=concepts/10010147.html>Computing methodologies</a></li>
    <li><a href=concepts/10003752.html>Theory of computation</a></li>
    <li><a href=concepts/10003456.html>Social and professional topics</a
        ></li>
    <li><a href=concepts/10003120.html>Human-centered computing</a></li>
    <li><a href=concepts/10003033.html>Networks</a></li>
    <li><a href=concepts/10002978.html>Security and privacy</a></li>
    <li><a href=concepts/10002951.html>Information systems</a></li>
    <li><a href=concepts/10002950.html>Mathematics of computing</a></li>
    <li><a href=concepts/10002944.html>General and reference</a></li>
</ul>
</tr>
  </table>
</body>
</html>
```

3.3.2 concepts/10010679.html

```
<html xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns# xmlns:rdfs=
    http://www.w3.org/2000/01/rdf-schema# xmlns:skos=http://www.w3.org
    /2004/02/skos/core# xmlns:dc=http://purl.org/dc/elements/1.1/>
<head>
    <meta http-equiv=Content-Type content=text/html; charset=Western (
        Windows 1252)>
    <title>The ACM Computing Classification System (CCS) :: Concept</
        title>
</head>
<body>
  <h1>The ACM Computing Classification System (CCS) :: Concept ::
      Temperature simulation and estimation :: 10010679</h1>
  <h2>Temperature simulation and estimation</h2>
  <h3>Alternative definitions:</h3>
  <ul>
    <li>temperature estimation</li>
    <li>temperature simulation</li>
  </ul>
  <h3>Broader:</h3>
  <ul>
    <li><a href=10010586.html>Thermal issues</a></li>
  </ul>
  <address>
    [<a href=../index.html>Main index</a>]
  </address>
</body>
</html>
```

Capítulo 4

Conclusões

Após a realização deste trabalho foi possível comprovar a utilidade de uma ferramenta como o Flex. Neste trabalho, a partir de um ficheiro SKOS imperceptível ao utilizador comum gerou-se um conjunto de páginas html com navegações entre as várias páginas, cuja forma de consulta é similar às consultas realizadas dezenas de vezes por dia numa navegação pela Internet. Mas a partir deste mesmo ficheiro, poder-se-ia ter gerado scripts de criação de uma base de dados, um qualquer outro formato que um outro programa estivesse à espera, ficheiros excel... As possibilidades são imensas. A realização deste trabalho permitiu também concluir que a estrutura xml inerente ao formato SKOS facilita bastante o desenho de um automatismo de processamento de texto.

Capítulo 5

Trabalho Futuro

- As relações "broader", "narrower" e "related" são bidirecionais o que quer dizer que quando um "Concept" tem uma ligação do tipo "narrower", pode-se inferir que o "Concept" apontado por essa ligação deverá ter uma ligação do tipo "broader" que aponte para o "Concept" inicial. No caso da "ACM Computing Classification System" os relações estavam completamente discriminadas o que gerou html perfeitamente navegável, contudo no ficheiro das localidades as ligações estavam descritas apenas no sentido ascendente, apenas existem ligações do tipo "broader" e não do tipo "narrower", o que gerou um html não tão navegável como o primeiro. Poderá implementar-se um método que a partir de uma relação infira a outra.
- Neste momento é necessário que exista uma pasta chamada concepts na mesma pasta do programa para que este funcione. Esta pasta deverá ser automaticamente criada pelo programa caso não exista.
- A página final neste momento contém uma formatação mínima, apenas uns headers e umas marcas. Seria interessante formatar a página resultante com bootstrap e CSS criando até menus para uma maior comodidade de navegação.

Bibliografia

[1] http://www.w3.org/TR/skos-reference/#semantic-relations