Integration of Biological Data Resources Using Image Object Keying

Nawaz Khan
School of Computing Science
Middlesex University, UK.
n.x.khan@mdx.ac.uk

Shahedur Rahman
School of Computing Science
Middlesex University, UK.
s.rahman@mdx.ac.uk

A. G. Stockman
Queen Mary College
University of London, UK
tonys@dcs.qmul.ac.uk

Abstract

This paper presents a novel approach to integrate biological data from multiple resources into a single page using image object keying. In this interactive approach, a gel electrophoresis protein spot is selected by the user which initiates the retrieval of corresponding 3D structure of the protein. It provides a set of operators to access and to collect elements content of the biological data resources for integration. The approach utilises a number of tools, namely, meta data extractor, mapping linker, dispatcher and result integrator for unifying the data collected from multiple data sources.

1. Introduction

The paper presents a novel approach for interoperable database search using image object keying. The paper describes: how gel electrophoresis image spot in source image can initiate data set searching from multiple biological resources and how it can integrate the results retrieved from different biological resources into a single page. This approach provides an alternative to the use of generic schema for database integration, by utilising loosely coupled schema which are less dependent on component data sources. It also coordinates the integration of component databases without actually depending on the top level schema/view model. This gives the scope to use laboratory based, target-specific component databases [2] which are relatively independent and autonomous and which will be able to interact with other molecular biology data sources for more meaningful information without participating into any database federation.

The behaviour of the prototype model depends on how it interacts with different biological data sources and the aim is to extract the required information with a single instance without writing any query scripts. The paper describes an approach to extract the information from the web and to automate it by using various associated tools. The unique feature of this approach is that even if the external data model is text or HTML based, the internal data model is always XML based, thus increasing the interoperability between the databases.

Proceedings of the 16th IEEE Symposium on Computer-Based Medical Systems (CBMS'03)
1063-7125/03 $17.00 © 2003 **IEEE**

2. Approach to database integration

The search scheme to obtain an integrated view of data set which is distributed over different data resources like Protein Data Bank (PDB: D_o), Genome Data Bank (GDB: D_g), and Online Mandelian Inheritance in Man (OMIM: D_o) is described in the following section.

The search scheme has employed the following tools to retrieve the required information from biological resources [2]: Mapping Linker, Meta Data Extractor, Image Feature Extractor, Dispatcher, and Result Integrator.

2.1 Mapping linker

The key feature here is that the researchers' will be able to configure their own choice of resource database mapping information according to their specific query. The Mapping Linker collects the resource mapping information from the component databases. This information is passed to the meta data extractor. The mapping linker uses Resource Description Framework (RDF) to process the meta data. The main feature of RDF [8] is to provide interoperability by adding semantics to the web resources. RDF describes the whole web page or part of the web page as resource and these resources are named by URI. Resource URI with their properties and values are used to define a RDF statement. For example, to define SWISS 2DPAGE resource database the following RDF statements are used to describe the meta data:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s=http://description.org/schmea/>
  <rdf:Description about=''http://www-lecb.ncifcrf.gov/cgi-bin/dbEngine/2dwgDB">
    <s:GelSpot>
     <rdf:Description about="http://www-lecb.ncifcrf.gov/cgi-bin/dbEngine/2dwgDB,getTableDataByID,WG00123''
       <rdf:type resource=http://description.org/schema/Proteins/>
       <v:ProteinID>        </v:ProteinID>
       <v:ProteinName>    </v:ProteinName>
     </rdf:Description>
   </s:GelSpot>
  </rdf:Description>
</rdf:RDF>
```

For some other conditions where a single web is referring to a collection of resources, for example, SWISS-2DPAGE gel electrophoresis protein spots which are referring to PDB, GDB and OMIM entry are described with RDF as follows:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s=''http://description.org/schmea/''>
 <rdf:Description about="http://www-lecb.ncifcrf.gov/cgi-bin/dbEngine/2dwgDB,getTableDataByID,WG00123''
    <s:protein>
       <rdf:Bag>
          <rdf:li  resource=''http://www.pdb.org'' />
          <rdf:li  resource=''http://www.gdb.org''/>
          <rdf:li  resource=''http://www.omim.og'' />
    </rdf:Bag>
   </s:protein>
  </rdf:Description>
</rdf:RDF>
```

And a HTML document containing RDF metadata is created as follows:

```
<html>
<head>
   <rdf:RDF
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:dc=''http://purl.org/metadata/dublin_core#''>
```

```
    <rdf: Description about = " ">
        <dc:SpotDetails>
            <rdf:Seq ID= "MatchedSpotDetails"
              <rdf:li>''http://www.pdb.org </rdf:li>
              <rdf:li>''http://www.gdb.org'' </rdf:li>
              <rdf:li> ''http://www.omim.og'' </rdf:li>
            </rdf:Seq>
        </dc:SpotDetails>
     </ref:Description>
    </rdf:RDF>
  </head>
</html>
```

2.2 Meta data extractor

Meta data extractor collects meta data which represents the format, structural details and links of the resource databases. It also extracts the contents of the elements and stores them in corresponding variables. For meta data extraction a wrapper module called LSXT is used [7]. LSXT is a data extraction and transformation tool for converting any data stream into XML. We have used LSX template which is used to extract elements contents by using *lsx:for-each* and *lsx:value-of select*. Then the values are stored in different variables, for example, *p, q* and *r*. LSX-T, which is complementary to XSLT, can read mapping linker HTML and converts it into XML. An example of converting mapping linker HTML containing resources into XML using LSXT is described below:

```
<lsx:transform xmlns:lsx=http://www.w3.org/lsx>
<SpotDetails>
  <lsx:template pattern-match="^Resource:  $p$;\s$q$;\s;$r$;\s\w+.">
  <lsx:details p="\w+" q="\w+" r="\w+"/>
  <Resource>
    <GDBResource>
      <lsx:value-of select="p"/>
     </GDBResource>
     <PDBResource>
       <lsx:value-of select="q"/>
     </PDBResource>
     <OMIMResource>
       <lsx:value-of select="r"/>
     </OMIMResource>
   </Resources>
 </SpotDetails>
</lsx:transform>
```

The converted XML for above LSXT template is given below:

```
<SpotDetails>
  <Resource>
        <GDBResource>''http://www.pdb.org'' </GDBResource>
        <PDBResource>''http://www.gdb.org''</PDBResource>
        <OMIMResource>''http://www.omim.og'' </OMIMResource>
   </Resource>
</SpotDetails>
```

2.3. Image feature extractor

This collects image data features based on their contents from the component database. This is used for image comparison and to understand the structural variances. An image content extractor module (M_I) is created to match protein spots between source and target gel

electrophoresis images. The module identifies the protein spot in the target image which lies on the same line of path as it is in the source image (*electrophoretic mobility* concept). A shape matching algorithm using *Generalized Hough Transform* and *Canny Edge Detection* method have been used to determine the shape variance. The set of spots in each gel electrophoresis is labelled and each spot is associated with a finite number of values, e.g., accession number of gene, protein and OMIM along with protein spot shape, mean value and positional vector. A class is formed using these labels and values. Each entry in the class is of the form (L,V) where L is a label and $V=\{v_1,.....,v_n\}$ is a set of values. Each v_i represents a value that could potentially be assigned to an element E, if label (E) matches L. In our case element E is a spot which corresponds to the specific 3-D structure of a protein. M_f, is used to look for the value v_i in order to begin the search for the corresponding element E. [3].

2.4. Dispatcher

The *Dispatcher* receives content based image descriptions from *Image feature extractor* and meta data from *Meta data extractor.* The *Dispatcher* then submits the operators to the individual resource databases to establish the link. For the search mechanism hyperlink is carried out by the Dispatch operator [4]. The basic functions of this search initiator, the dispatcher, are: *i.* split the search operators; *ii*. allocate them to multiple bioinformatics sources; *iii.* determine each search operator as a sub-plan of the total output; *iv*. create dynamic memory to hold the subset of the output for further integration. To link the selected gel electrophoresis spot with the bioinformatics resources, it is necessary to directly link to the requested page p in order to optimise the hyperlink data searching. For example, if data set d is distributed over a number of biological resources, D_p, D_g *and* D_o, then to retrieve d, the following steps are performed: *i*. access to the D_p, D_g and D_o resources*; ii.* retrieve the required pages p_p, p_g, p_o from D_p, D_g *and* D_o; *iii*. access the required set of elements e_p, e_g and e_o from the pages p_p, p_g, p_o respectively and *iv*. then pass the elements of the pages to the result integrator to embed the elements into a single page p. The steps are shown in figure 1. The overall objective of the dispatcher is to apply a set of search operators O to the respective data resource R described in mapping linker and let $O_i(R_i)$ ($1=<i=<n$, where n is a finite value) be a set of derived facts related to the overall search result.

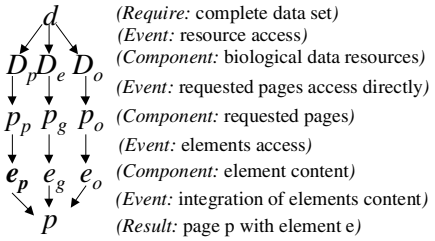


Figure 1: Events and Components for searching the element content

To extract the element contents the following code is used.

```
public static String getText(String uriStr) {
        final StringBuffer buf = new StringBuffer();
        try {
        HTMLDocument doc = new HTMLDocument() {
               public HTMLEditorKit.ParserCallback getReader(int loc) {
                    return new HTMLEditorKit.ParserCallback() {
                public void handleText(char[] data, int loc) {
                           buf.append(data);
                           buf.append('\n');}};};
            URL url = new URI(uriStr).toURL();
```

```
        URLConnection conn = url.openConnection();
        Reader rd = new InputStreamReader(conn.getInputStream());
        EditorKit kit = new HTMLEditorKit();
        kit.read(rd, doc, 0);
    } catch (MalformedURLException e) {
    } catch (URISyntaxException e) {
    } catch (BadLocationException e) {
    } catch (IOException e) {
    } return buf.toString(); }
```

2.5. Result integrator

The Result Integrator captures all the results from individual resource databases. It then presents the results in an integrated form to the user along with images and other related local information stored in the component databases. Document Object Model (DOM) has been used which is an application programming interface (API) for valid HTML and well-formed XML documents. It defines the logical structure of the documents. A converter module is then used to convert the DOM file into XML documents for interoperability. A segment of converter module is given below:

```
try
{source = new DOMSource(doc);
 File file = new File(filename);
 Result result = new StreamResult(file);
Transformer xformer= TransformerFactory.newInstance().newTransformer();
 xformer.transform(source, result);
 } catch (TransformerConfigurationException e) {
  } catch (TransformerException e) { }  }
```

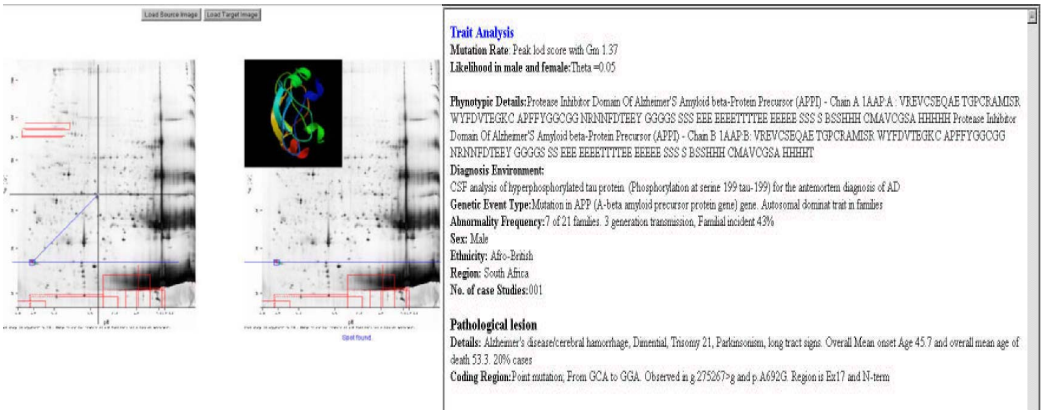
Finally, an Extensible Stylesheet Language Translator (XSLT) template is used to extract the XML elements from the converted XML files.

3. Integration of biological data sources: An example

The implementation of the integration approach is carried out by using tools described in the above sections. DOM [9] interface is used which provides a set of API calls for accessing the content of the documents; a special wrapper designed for DOM-compliant data sources exports this information to the dynamic buffer (a virtual table) for each such API call [5]. The input parameters for the DOM calls are *accession numbers* of the biological resources which are obtained by scanning through the RDF data. To illustrate the process, Alzheimer disease is taken as an example. The protein spot (*APP1*) for Alzheimer's disease is selected from the gel electrophoresis image (Fig 2a). The spot position is then matched in the target image and the 3D structural protein image for *APP1* is retrieved by matching the image content elements using image content extractor (fig 2a). Now the *Dispatcher* aims to dispatch the searching operators individually to the respective data resources for unifying them into a single page. For example, the following individual resources along with their operators (collected from RDF) are sent to the respective databases for Alzheimer's disease details and to correlate the gel protein spot to the *geno* and *phenotypic* information. DOM interfaces are used for the following resources and operators to traverse along the contents.

Resource r1← http://us.expasy.org/cgi-bin/niceprot	Operator s1.PI=P05067
Resource r2← http://us.expasy.org/cgi-bin/blast.PI?	Operator s2 sequence=P05067
Resource r3← http://www.rcsb.org/pdb/cgi/explore.cgi?	Operator s3 pdbId=1AAP
Resource r4← http://www.gdb.org/gdb-bin/genera/accno?	Operator s4 accessionNum=GDB:119692
Resource r5← http://www.ncbi.nlm.nih.gov/htbin-post/Omim	Operator s5 dispmim=104300

Figure 2(a) shows an interface where a source gel electrophoresis image is loaded and the spot for *APP1* protein is selected. A target image is then loaded with spot in the same position and the contents of the spots are matched with RDF document which would enable to find out the resources and operators in order to search for other details. Figure 2(b) is the single page of HTML which consists of the data elements retrieved from multiple databases. These data elements are retrieved and unified when the *Dispatcher* dispatches the searching and *the Result Integrator* integrates the results.



a **b**

Figure 2: (a) *APP1* protein spot matching and (b)Element collection and integration.

4. Conclusion

In this paper we have emphasised how biological data sources can be linked together to present an integrated result which identifies a specific protein spot selected interactively by the user. In this approach we have initiated the search mechanism by selecting an image object. This approach is novel because it does not depend on any script writing to initiate the search. The paper also focuses on how to describe a complete data set (from gel protein spot to phenotypic details and disease state of a protein) dynamically without using any data federation or data warehouse approach for database integration.

5. References

[1]Khan, N., Rahman, S and Clarkson, G. T.; (2001) An approach to develop human gene disorder database for intelligent variance analysis of genes and its products. 2001, 12th International workshop on Database and Expert System (DEXA, 2001). Germany, Munich. Proc. IEEE Computer society press.

[2] Khan, N and Rahman, S; (2001) A conceptual object modelling of gene mutation data. 2001, German Conference of Bioinformatics, GCB01. Germany, Proc.

[3]Khan, N., Stockman, A.G. and Rahman, S. (2002) A Cooperative Environment for Genetic Variance Analysis Using Component Database for Database Integration. 15th IEEE Conference on Medical Based Systems (CMBS, 2002). Proc. IEEE Computer Society Press. Slovania, June.

[4] Khan, N. and Rahman, S (2003) A New Approach to Detect Similar Proteins from 2D Gel Electrophoresis Images. Proc. IEEE 3rd Symposium on Bioinformatics and Bioengineering.

[5] Kemper, A and Wiesner, C (2001) Hyper Queries: Dynamic Distributed Query Processing on the internet, Proc. of VLDB Conference.

[6] Manolescu, I, Florescu, D, Kossmann, D, Xhumari, F and Olteanu, Dagora, (2000) Living with XML and Relational, Proc. VLDB conference.

[7] Wong, R.K. and Shui, W.M., (2001), Utilizing Multiple Bioinformatics Information Sources: An XML Database Approach, Proc. IEEE 3rd Symposium on Bioinformatics and Bioengineering.

[8] http://www.w3.org Resource Description Framework (RDF) Model and Syntax Specification.

[9] http://www.w3.org/TR/REC-DOM-Level-1, DOM Specification Level 1