An XML application for genomic data interoperation

Kei-Hoi Cheung[1], Yang Liu[2], Anuj Kumar[3], Michael Snyder[2,3], Mark Gerstein[2], Perry Miller[1,3]
[1]Center for Medical Informatics, Department of Anesthesiology [2] Department of Molecular Biophysics and Biochemistry, [3]Department of Molecular, Cellular and Developmental Biology, Yale University, New Haven, CT 06520, USA
{kei.cheung, anuj.kumar, michael.snyder, mark.gerstein, perry.miller}@yale.edu

Abstract

As the eXtensible Markup Language (XML) becomes a popular or standard language for exchanging data over the Internet/Web, there are a growing number of genome Web sites that make their data available in XML format. Publishing genomic data in XML format alone would not be that useful if there is a lack of development of software applications that could take advantage of the XML technology to process these XML-formatted data. This paper illustrates the usefulness of XML in representing and interoperating genomic data between two different data sources (Snyder's laboratory at Yale and SGD at Stanford). In particular, we compare the locations of transposon insertions in the yeast DNA sequences that have been identified by BLAST searches with the chromosomal locations of the yeast open reading frames (ORFs) stored in SGD. Such a comparison allows us to characterize the transposon insertions by indicating whether they fall into any ORFs (which may potentially encode proteins that possess essential biological functions). To implement this XML-based interoperation, we used NCBI's "blastall" (which gives an XML output option) and SGD's yeast nucleotide sequence dataset to establish a local blast server. Also, we converted the SGD's ORF location data file (which is available in tab-delimited format) into an XML document based on the BIOML (BIOpolymer Markup Language) standard.

1. Introduction

With the growing use of the Web, large quantities of biological data have been made accessible to the scientific community through many genome Web sites such as the National Center for Biotechnology Information (NCBI) Web site (http://www.ncbi.nlm.nih.gov/) and the Protein Data Bank (PDB) Web site (http://www.rcsb.org/pdb/). The Web has widely been accepted because it is an Internet-based standard that has been incorporated into multiple platforms (e.g., Unix, Windows, and Mac). Web browsers such as Netscape and Internet Explorer (IE) are platform-independent and are easy to use. They provide for the user the capability of browsing through a large set of data graphically on their local computers. This graphical capability is made possible by the HyperText Markup Language (HTML). Another important feature of the Web is that it allows multiple HTML documents resided on different Web servers to be linked to one another through hypertext links. Such hypertext links have revolutionized the way information (stored in remote databases) can be linked over the Internet. This level of database inter-connectivity has already proven very useful since it allows the user to navigate data from one Web site to another related Web site very easily. According to Karp [1], however, this is not the "Holy Grail" of genomic data interoperation. There are two problems associated with this hypertext linking approach.

1. **Item-by-item linking.** One problem is that the user has to click on the link one at a time in order to retrieve related information. This will be a time-consuming and tedious method to collect related information if the number of links involved is great, thereby slowing data collation prior to analysis.
2. **Fixed linked fields.** The other problem occurs when we attempt to establish links to external data sources. These external links restrict how we can access related data. For example, some genome databases allow their data entries to be linked via accession numbers (unique object/record identifiers). However, if these numbers are not available in the public interface, there will be no way to establish links using other fields (such as gene names or gene symbols).

Despite its hyperlink capability, the HTML is designed mainly for data display purposes. It is not suitable for large-scale machine processing. To address this, many genome sites have distributed large datasets as flat files (e.g., tab-delimited files). Researchers can then download these files and process them by custom programs. According to [2], this flat-file approach is very limited, because it lacks such abilities as referencing, controlled vocabulary, and constraints. Often fields are ambiguous and their contents are contextual. In other words, the programmer has to manually interpret the semantics. This hinders the use of flat files by programs without human interaction. To fully automate genomic-scale data interoperation, we need a data representation format that not only separates the semantic content from the display content, but it also allows computer programs to process the semantic part efficiently.

The eXtensible Markup Language (XML) has emerged as a popular format (both human and machine readable) for exchanging information over the Web.

XML was designed to overcome the limitations of HTML and flat files as described previously. It is derived from the Standard Generalized Markup Language (SGML), the international standard for defining descriptions of the structure and content of different types of electronic documents. XML documents are self-describing. A set of user-defined tags can be created for one or many XML documents. Syntactic and semantic rules can be defined for these tags in the form of Document Type Definitions (DTD). In general, the XML tags are used to identify different types of hierarchically related elements in the document, with the possibility of referencing and recursion. Besides its use in data publishing, XML gives the means for defining strongly structured documents so that computer programs can easily navigate through them and access relevant pieces of information. Another advantage of using XML is that there is a large body of XML-related software tools and technologies including Document Object Model (DOM) and eXtensible Stylesheet Language (XSL) that are available in the public domain.

There has been an increasing use of XML in the genome community. Recently, we have seen a growing number of genome sites that distribute data in XML format. Among these are NCBI, PDB, and Gene Ontology (http://www.geneontology.org/). In addition, a number of XML-related standards have been proposed for representing different types of biological data. Among them are MAML (http://www.ncbi.nlm.nih.gov/geo/maml/index.cgi) and GEML (http://www.geml.org/), which are XML-based languages for describing gene expression data; BIOML (http://www.bioml.com/BIOML/) that describes biopolymers including genes and proteins; and BSML (http://www.labbook.com/products/xmlbsml.asp) that describes DNA sequence data.

This paper describes how to use XML to represent and interoperate the yeast data that have been produced at two different sites: Snyder's Lab at Yale and the Saccharomyces Genome Database (SGD) [3] at Stanford. The paper is organized as follows. Section 2 will give an overview of the yeast genomic project to which our XML approach is applied. This section will also describe the computer programs used to process the data in different stages. In Section 3, we will describe our XML approach to interoperating the yeast datasets of interest. Also, some examples will be given. Section 4 will provide some discussion of how to improve and extend our work. Finally, we will give the conclusion in Section 5.

2. Application Domain and Data Processing

Fig. 1 gives an overview of the yeast genomic project to which our XML approach is applied. This project involves a large-scale functional analysis of the yeast genome by transposon mutagenesis [4]. The data generated from this project are stored in a Web-accessible database—TRIPLES [5]. This research project generates a large collection of mutant yeast strains or DNA sequences (represented by short dark lines in Fig. 1), each strain carrying a transposon insertion (represented pictorially by an inverted triangle) at a defined site within the yeast genome.

These mutant strains are subsequently used in a variety of functional studies, enabling the analysis of gene expression, disruption phenotypes, and protein localization [6]. This strain collection is maintained in 96-well format. Each strain is assigned a unique ID based upon its position within a 96-well storage plate. For example, strain "V108B6" is stored in plate 108 at position B6. The prefix "V" indicates that this strain carries a transposon insertion within a region of the yeast genome expressed during vegetative growth. Yeast cells propagate vegetatively when provided with sufficient nutrients; under appropriate conditions of starvation, however, yeast cells undergo meiosis and spore formation. Strains carrying a transposon insertion affecting a gene whose expression is induced during this sporulation process are named with an "M" (meiotic) prefix. This ID designation is useful in tracking given strains during subsequent analysis steps. As described below, number of computer programs are used to identify a genomic region (e.g., a gene) disrupted by a transposon insertion within each strain.

- **Sequencing**. An initial step of the project is to sequence the DNA samples (yeast mutant strains) collected. Automatic DNA sequencers such as the ones manufactured by Applied Biosystems support high throughput sequencing. Following high-throughput automated sequencing, DNA sequence data is typically output as chromatograms (trace signals) that must be subsequently converted into nucleotide sequence. These chromatogram data sets are stored in binary files ("chromat" files). In our case, each chromat file contains DNA sequence data for a given strain. These files are named according to the strain ID described previously. We process these chromat files with the PHRED-PHRAP package [7, 8] to produce nucleotide sequences (clone sequences). In addition, we have configured PHRED-PHRAP to remove vector sequence as well as the transposon sequence itself from each clone sequence. Occasionally, transposon sequence are missed by PHRED-PHRAP due to sequencing errors. These errors generate DNA sequence data that imperfectly matches the pre-specified transposon sequence. Often, these sequencing errors are minor; manual inspection is usually sufficient to identify transposon sequence in these cases. To address this problem, a script was written to scan the output of PHRED-PHRAP for varying patterns of this transposon sequence. Specifically, the sequence data are scanned for a region of 10 nucleotides corresponding to the extreme 5' end of the transposon. This automatic "pattern-matching" is helpful in reducing manual labor, thereby streamlining data processing.
- **Sequence homology searching**. Following PHRED-PHRAP processing, the resulting DNA sequence data is searched against the yeast genome as a means of identifying the genomic site of transposon insertion. For this purpose, sequences are submitted for BLAST [9] searches. We have implemented a local BLAST server by using the "BLASTALL" program from NCBI and the yeast nucleotide sequence sets from SGD. We have written scripts to allow multiple sequence files (each

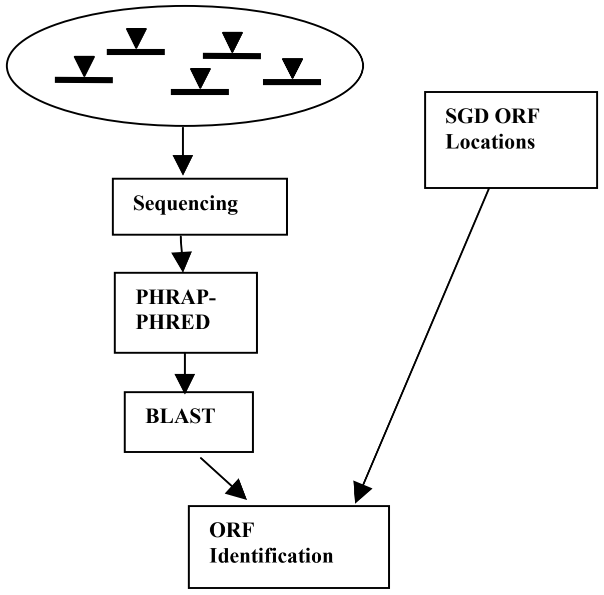

Fig. 1. Yeast transposon insertions and ORF identification.

of which can contain multiple sequences) to be submitted for BLAST searches. This batch submission is necessary in order to analyze the large volume of sequence data generated in a typical genome project.

- **Identification of ORFs**. The BLAST results returned from SEARCH-LAUNCHER are processed automatically to identify both the exact site of transposon insertion within the yeast genome as well as open reading frames disrupted by this insertion event. Based on the BLAST output (sequence alignments), the chromosomal coordinate of transposon insertion is calculated. The resulting insertion coordinate is then compared with the start and end chromosomal coordinates of all annotated ORFs recorded in SGD.

3. Implementation of XML Interoperation

The datasets that we attempt to interoperate using XML involve the BLAST output data and the ORF location data obtained from SGD (ftp://genome-ftp.stanford.edu/pub/yeast/tables/ORF_Locations/ORF_table.txt). As described previously, we used the "BLASTALL" program provided by NCBI to perform local BLAST searches. This program allows the BLAST output to be formatted in XML. The DTD for the BLAST XML output can be obtained via the following: ftp://ncbi.nlm.nih.gov/blast/documents. This XML structure was derived from the ASN.1 structure of BLAST and is still experimental. The input to the BLASTALL program is a sequence file containing the individual sequences. In our case, each sequence file represents a 96-well plate and therefore consists of 96 sequences, each of which is identified by a strain (or clone) ID as described previously. The BLASTALL program will produce an XML document as output for each sequence (the sequence strain ID is used to name the XML document). To facilitate large-scale processing, we have written scripts to allow a batch of sequence files to be submitted for BLAST searches.

The ORF location data file provided by SGD is available in tab-delimited format. In order to make our data interoperation truly XML-based, we converted this data file into an XML document. Instead of defining an arbitrary XML structure, we used BIOML (BIOpolymer Markup Language) as a guide to implement the conversion. In general, BIOML describes information about biopolymers (e.g., genes and proteins) including the chromosomal locations of DNA sequences. ORFs can be considered a specific type of DNA sequences (the ones that encode proteins). To make this fact explicit, we modified the definition (DTD) of BIOML slightly to include ORF as a new element. In the following, some examples are provided to illustrate the BLAST XML output and the SGD ORF location data in XML format. Also, we describe how to interoperate these XML documents.

3.1. Blast XML Output

This section gives examples to illustrate the XML-formatted output of BLAST for two different strain sequences: one has matches with yeast genome sequence and the other has no match.

<u>A. Match Example</u>

The XML example below illustrates the BLAST matches of the input (query) sequence "V97A1". This BLAST output includes the following: which BLAST program (and what version) is used (e.g., blastn 2.1.3 is used for nucleotide sequence searching); which genome sequence database is used for matching the query or input sequence(s); description of the query sequence (e.g., the name of the sequence); the parameters used in performing the BLAST search (in this example, the filter "D", which stands for DUST, is used to filter the query sequence); descriptions of the matches or hits; and statistics. In general, a BLAST search can result in multiple hits in different regions of the target (yeast) genome. Each hit is characterized by a set of High-scoring Segment Pairs (HSPs) that include pairs of aligned sequences with the corresponding alignment scores.

```
<?xml version="1.0"?>
<!DOCTYPE BlastOutput PUBLIC "-//NCBI//NCBI BlastOutput/EN" "NCBI_BlastOutput.dtd">
<BlastOutput>
  <BlastOutput_program>blastn</BlastOutput_program>
  <BlastOutput_version>blastn 2.1.3 [Apr-11-2001]</BlastOutput_version>
  <BlastOutput_reference>
     ~Reference: Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer, ~Jinghui
     Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997), ~&quot;Gapped
     BLAST and PSI-BLAST: a new generation of protein database
     search~programs&quot;,  Nucleic Acids Res. 25:3389-3402.
  </BlastOutput_reference>
  <BlastOutput_db>../Blast/chr_all.nt</BlastOutput_db>
  <BlastOutput_query-ID>lcl|QUERY</BlastOutput_query-ID>
  <BlastOutput_query-def>V97A1</BlastOutput_query-def>
  <BlastOutput_query-len>346</BlastOutput_query-len>
  <BlastOutput_param>
    <Parameters>
      <Parameters_expect>10</Parameters_expect>
      <Parameters_include>0</Parameters_include>
      <Parameters_sc-match>1</Parameters_sc-match>
      <Parameters_sc-mismatch>-3</Parameters_sc-mismatch>
      <Parameters_gap-open>5</Parameters_gap-open>
      <Parameters_gap-extend>2</Parameters_gap-extend>
      <Parameters_filter>D</Parameters_filter>
    </Parameters>
  </BlastOutput_param>
  <BlastOutput_iterations>
    <Iteration>
      <Iteration_iter-num>1</Iteration_iter-num>
      <Iteration_hits>
        <Hit>
          <Hit_num>1</Hit_num>
          <Hit_id>ref|NC_001147|</Hit_id>
          <Hit_def>[org=Saccharomyces cerevisiae] [strain=S288C] [moltype=genomic]
[chromosome=XV]</Hit_def>
          <Hit_accession>NC_001147</Hit_accession>
          <Hit_len>1091284</Hit_len>
          <Hit_hsps>
                <Hsp>
                  <Hsp_num>1</Hsp_num>
                  <Hsp_bit-score>686.389</Hsp_bit-score>
                  <Hsp_score>346</Hsp_score>
                  <Hsp_evalue>0</Hsp_evalue>
                  <Hsp_query-from>346</Hsp_query-from>
                  <Hsp_query-to>1</Hsp_query-to>
                  <Hsp_hit-from>1089180</Hsp_hit-from>
                  <Hsp_hit-to>1089525</Hsp_hit-to>
                  <Hsp_pattern-from>0</Hsp_pattern-from>
                  <Hsp_pattern-to>0</Hsp_pattern-to>
                  <Hsp_query-frame>1</Hsp_query-frame>
                  <Hsp_hit-frame>-1</Hsp_hit-frame>
                  <Hsp_identity>346</Hsp_identity>
                  <Hsp_positive>346</Hsp_positive>
                  <Hsp_gaps>0</Hsp_gaps>
                  <Hsp_align-len>346</Hsp_align-len>
                  <Hsp_density>0</Hsp_density>
                  <Hsp_qseq>
                     TATTGCAGCAGTGATGAGGACAGCGACACGTGCATTCATGGTAG
                     TGCTAATGCCAGTACCAATGCGACTACCAACTCCAGCACTAATGC
                     TACTACCACTGCCAGCACCAACGTCAGGACTAGTGCTACTACCAC
                     TGCCAGCATCAACGTCAGGACTAGTGCGATTACCACTGAAAGTA
                     CCAACTCCAGCACTAATGCTACTACCACTGCCAGCACCAACGTCA
                     GGACTAGTGCTACTACCACTGCCAGCATCAACGTCAGGACTAGT
                     GCGACTACCACTGAAAGTACCAACTCCAACACTAGTGCTACTACC
                     ACCGAAAGTACCGACTCCAACACTAGTGCTACTA
                  </Hsp_qseq>
                  <Hsp_hseq>
                     TATTGCAGCAGTGATGAGGACAGCGACACGTGCATTCATGGTAG
                     TGCTAATGCCAGTACCAATGCGACTACCAACTCCAGCACTAATGC
                     TACTACCACTGCCAGCACCAACGTCAGGACTAGTGCTACTACCAC
                     TGCCAGCATCAACGTCAGGACTAGTGCGATTACCACTGAAAGTA
                     CCAACTCCAGCACTAATGCTACTACCACTGCCAGCACCAACGTCA
                     GGACTAGTGCTACTACCACTGCCAGCATCAACGTCAGGACTAGT
                     GCGACTACCACTGAAAGTACCAACTCCAACACTAGTGCTACTACC
                     ACCGAAAGTACCGACTCCAACACTAGTGCTACTA
                  </Hsp_hseq>
                  <Hsp_midline>
                     ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
                     ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
                     ||||||||||||||||||||||||||||||||||||||||||||||
                  </Hsp_midline>
                </Hsp>
                      .
                      .
            </Hit_hsps>
          </Hit>
                      .
                      .
        </Iteration_hits>
        <Iteration_stat>
          <Statistics>
            <Statistics_db-num>17</Statistics_db-num>
            <Statistics_db-len>12156302</Statistics_db-len>
            <Statistics_hsp-len>0</Statistics_hsp-len>
            <Statistics_eff-space>4.01149e+09</Statistics_eff-space>
            <Statistics_kappa>0.710605</Statistics_kappa>
            <Statistics_lambda>1.37407</Statistics_lambda>
            <Statistics_entropy>1.30725</Statistics_entropy>
          </Statistics>
        </Iteration_stat>
      </Iteration>
    </BlastOutput_iterations>
</BlastOutput>
```

B. No Match Example

The example below illustrates that the input sequence "V97A5" has no match/hit in the yeast genome sequence. It is obvious in the XML output that there are no hit descriptions. Also shown in the example (near the bottom) is the following element-value pair: "<Iteration_message>No hits found</Iteration_message>".

```
<?xml version="1.0"?>
<!DOCTYPE BlastOutput PUBLIC "-//NCBI//NCBI BlastOutput/EN" "NCBI_BlastOutput.dtd">
<BlastOutput>
  <BlastOutput_program>blastn</BlastOutput_program>
  <BlastOutput_version>blastn 2.1.3 [Apr-11-2001]</BlastOutput_version>
  <BlastOutput_reference>
     ~Reference: Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer, ~Jinghui
     Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997), ~&quot;Gapped
```

```
      BLAST and PSI-BLAST: a new generation of protein database
      search~programs&quot;, Nucleic Acids Res. 25:3389-3402.
  </BlastOutput_reference>
  <BlastOutput_db>../Blast/chr_all.nt</BlastOutput_db>
  <BlastOutput_query-ID>lcl|QUERY</BlastOutput_query-ID>
  <BlastOutput_query-def>V97A5  </BlastOutput_query-def>
  <BlastOutput_query-len>32</BlastOutput_query-len>
  <BlastOutput_param>
    <Parameters>
      <Parameters_expect>10</Parameters_expect>
      <Parameters_include>0</Parameters_include>
      <Parameters_sc-match>1</Parameters_sc-match>
      <Parameters_sc-mismatch>-3</Parameters_sc-mismatch>
      <Parameters_gap-open>5</Parameters_gap-open>
      <Parameters_gap-extend>2</Parameters_gap-extend>
      <Parameters_filter>D</Parameters_filter>
    </Parameters>
  </BlastOutput_param>
  <BlastOutput_iterations>
    <Iteration>
      <Iteration_iter-num>1</Iteration_iter-num>
      <Iteration_stat>
        <Statistics>
          <Statistics_db-num>17</Statistics_db-num>
          <Statistics_db-len>12156302</Statistics_db-len>
          <Statistics_hsp-len>0</Statistics_hsp-len>
          <Statistics_eff-space>2.30966e+08</Statistics_eff-space>
          <Statistics_kappa>0.710605</Statistics_kappa>
          <Statistics_lambda>1.37407</Statistics_lambda>
          <Statistics_entropy>1.30725</Statistics_entropy>
        </Statistics>
      </Iteration_stat>
      <Iteration_message>No hits found</Iteration_message>
    </Iteration>
  </BlastOutput_iterations>
</BlastOutput>
```

3.2. SGD location data example

The example below illustrates how the SGD location data are represented using the BIOML syntax. As mentioned previously, we have extended the BIOML structure to include open reading frames (orf's). This new element is modeled in a very similar way to "locus" that is included in the original BIOML DTD. Both "locus" and "orf" are modeled as sub-elements of "chromosome" that, in turns, is a sub-element of "organism". The "orf" element has the same sub-elements (e.g., gene, dna and db_entry) as locus has. Both elements have almost the same set of attributes (e.g., start and end chromosomal coordinates) except that "orf" uses an additional attribute "introns" to indicate the number of introns (if any) within the open reading frame. We introduced this attribute mainly because there is an "introns" column in the source file. Also notice in the example that there is a "file" element that is used to describe the data source, including the URL through which the file can be downloaded and the description of each column.

```
<?xml version="1.0"?>
<!DOCTYPE bioml SYSTEM "bioml.dtd">
<bioml label="SGD chromosomal location data">
    <organism label="Saccharomyces cerevisiae">
                .
                .
        <chromosome label="V" number="5">
                .
                .
            <orf label="YER179W"  start="548416" end="549512" introns="1">
              meiosis-specific protein related to RecA and Rad51p. Dmc1p colocalizes with
              Rad51p to discrete subnuclear sites in nuclear spreads during mid prophase,
              briefly colocalizes with Zip1p, and then disappears by pachytene
                <db_entry label="orf" format="SGD" entry="S0000981">
                </db_entry>
                <gene label="dmc1">
                </gene>
                <dna label="YER179W">
                    <exon label="exon 1" start="1" end="132">
                    </exon>
                    <exon label="exon 2" start="225" end="1097">
                    </exon>
                </dna>
            </orf>
                .
                .
        </chromosome>
                .
                .
    </organism>
    <file label="SGD ORF LOC" URL="ftp: // genome-ftp.stanford.edu / pub / yeast / tables
    / ORF_Locations / ORF_table.txt" format="TAB-LIMITED">
      Table of Saccharomyces cerevisiae ORF Information. This table was produced by the
      Saccharomyces Genome Database project (http://genome-www.stanford.edu/). This is
      a tab-delimited file. The columns do not all line up when viewed with a text editor or
      word processor. However the tabs allow this file to be imported into a spreadsheet
      program without any changes. The table includes all Open Reading Frames (ORF)
      given a name by the systematic sequencers of the yeast genome. Unless experimental
      evidence or strong sequence similarity exists an ORF must encode a protein of 100
      amino acids or great to be given a systematic named. Some small ORFs are surely
      missing from the current list. Information included, 1) ORF Standard Name 2) SGDID
      for the ORF 3) Gene Name (if available) 4) Chromosome 5) Starting nucleotide within
      the currently know chromosomal sequence 6) Ending nucleotide within the currently
      know chromosomal sequence 7) Number of introns contained within the ORF 8) Exon
      coordinates where 1 is the first nucleotide of the ORF 9) Brief Description of gene
      product Note, ORFs encoded on the complement strand relative to the systematic
      sequence submitted to the public databases will have a starting nucleotide number larger
      than the ending nucleotide number. Also all ORFs will include the stop codon. Please
      report errors or suggestions of how this table can be more useful to Mike Cherry
      (cherry@genome.stanford.edu).
    </file>
</bioml>
```

3.3. Interoperation

We wrote a PERL program that uses the Document Object Model (DOM) module to interoperate the XML documents for both the BLAST output and SGD location data. Using DOM, the XML documents are mapped into a tree structure in memory. DOM also provides a number of methods to access different parts of the tree efficiently and easily (e.g., accessing elements by their names).

For the BLAST output that involves multiple hits (HSPs), our program is designed to choose the first HSP with the following two conditions:

1. A query sequence whose start or end position is one (indicated by **Hsp_query-from** or **Hsp_query-to**).

2. An e-value (the value of the **Hsp_evalue** element) that is equal to or smaller than a threshold value.

Once an HSP that satisfies the above conditions is found, we can determine the orientation (*ascending* or *descending*) and the chromosomal position of the transposon insertion by comparing the start and end positions (specified by **Hsp_query-from** and **Hsp_query-to**) of the query sequence with those (specified by **Hsp_hit-from** and **Hsp_hit-to**) of the hit sequence. If the value of **Hsp_query-from** is equal to one, the orientation is *ascending* and the insertion position is obtained from the **Hsp_hit-from** element. Otherwise, the orientation is *descending* and the insertion position is obtained from the **Hsp_hit-to** element. Given the chromosomal position of the transposon insertion, our program compares it with the start and end chromosomal coordinates of the exons of the ORFs contained in the SGD XML document (some ORFs have multiple exons). If it falls within an exon, a match is reported by the program.

4. Discussion

By representing the SGD data in XML, we have noted a number of advantages over the use of flat files. First of all, the order of the XML elements is insignificant. This makes the code easier to maintain. In the flat file approach, a change in the order of the columns would require code modification. Using the BIOML structure, the information about a chromosome (e.g., its label) is stored only once. The same piece of information is stored redundantly in the flat file format. Also, XML tends to be more efficient in accessing data in comparison with the flat file approach. When comparing the insertion coordinate (within a particular chromosome) obtained in the BLAST output with the coordinates of the exons of the ORFs, we have to access each ORF sequentially using the flat file approach. Using XML instead, we can access the chromosome directly and then iterate through the ORFs within that chromosome.

We adopted a relatively simple rule (the use of a threshold value) to scan the BLAST search results for sequence hits. Using this strategy, we may miss functionally significant matches in BLAST database searches. Programs such as BEAUTY [10] provide additional information (e.g., the locations of local hits and any annotated domains) based on BLAST search post-processing. Given such additional information, we may be able to identify more hits.

We characterized the transposon insertions by detecting their chromosomal locations in the yeast genome and identifying the known open reading frames (ORFs) disrupted by these insertion events. There are more ways to characterize these transposon insertions. For example, the insertions can be characterized as "in-frame" or "out-of-frame". Also, we have not discussed how to characterize those insertions that do not fall into any known ORFs. In this case, we can determine if they are within any ORFs that have not been annotated previously. We call such ORFs non-annotated open reading frames (NORFs). There are programs such as NCBI's ORF finder (http://www.ncbi.nlm.nih.gov/gorf/gorf.html) that can identify ORFs by scanning the start and stop codons that are embedded in a nucleic sequence. As described in [11], we have developed a program (ORFSEEK) to identify NORFs for the yeast genome based on the transposon insertions.

The XML documents (BLAST output and SGD location data) are processed and interoperated using DOM. This approach may yield poor performances when dealing with large XML documents. For processing large XML documents, we may use alternative XML technologies such as the Simple API for XML (SAX).

We wrote a PERL program to parse the SGD data file (a tab-delimited flat file) and convert it into an XML document. However, this approach will not scale if the XML conversion needs to be applied to a large number of files that are structured differently. In this case, each file would require a separate parsing program. To make this conversion process scalable, we have explored a metadata approach [12] that uses a central metadata repository to represent and store the structural mapping rules between the source files and the target XML documents. Then a single generic program can be written to process these rules to perform the XML conversion. Rules can be added or modified by simply editing the metadata without the need to change the conversion program.

5. Conclusion

We have demonstrated how to use XML to represent and interoperate data for a yeast genomic project involving transposon insertions. The results of this interoperation include location of the transposon insertions within the yeast genome including those that fall into the previously identified open reading frames. XML is self-describing and hierarchical. It also provides a machine-readable format for capturing data semantics. Our XML approach involved using the NCBI's BLASTALL program that is capable of producing XML output and converting the SGD' s ORF location dataset into XML based on the extension of BIOML. We also used an XML-related technology, DOM, to process the XML-formatted data. We discussed how our work could be improved and extended. In summary, our work lends support to the idea of using XML to distribute and exchange genomic data over the Web.

Acknowledgements

This work was supported in part by NIH grants G08 LM05583 from the National Library of Medicine, R01 CA77808, and 1K25HG02378-01. The authors would like to thank Kim Worley at the Baylor College of Medicine (BCM) and Wayne Matten at NCBI for pointing to the XML output of BLASTALL.

References

[1] Karp, P., A Strategy for Database Interoperation. Journal of Computational Biology, 1995. **2**(4): p. 573-586.

[2] Achard, F., G. Vaysseix, and E. Barillot, XML, bioinformatics and data integration. Bioinformatics, 2001. **17**(2): p. 115-125.

[3] Cherry, J., C. Adler, C. Ball, S. Chervitz, S. Dwight, E. Hester, Y. Jia, G. Juvik, T. Roe, M. Schroeder, S. Weng, and D. Botstein, SGD: Saccharomyces Genome Database. Nucleic Acids Res., 1998. **26**(1): p. 73-79.

[4] Burns, N., B. Grimwade, P. Ross-Macdonald, E. Choi, K. Finberg, G. Roeder, and M. Snyder, Large-scale analysis of gene expression, protein localization, and gene disruption in Saccharomyces cerevisiae. Genes Dev., 1994. **8**(9): p. 1087-1105.

[5] Kumar, A., K. Cheung, P. Ross-Macdonald, P. Coelho, P. Miller, and M. Snyder, TRIPLES: a database of gene function in S. cerevisiae. Nucleic Acids Research, 2000. **28**(1): p. 81-84.

[6] Ross-Macdonald, P., P. Coelho, T.R. T, S. Agarwal, A. Kumar, R. Jansen, K. Cheung, A. Sheehan, D. Symoniatis, L. Umansky, M. Heidtman, K. Nelson, H. Iwasaki, K. Hager, M. Gerstein, P. Miller, G.R. GS, and M. Snyder, Large-Scale Analysis of the Yeast Genome by Transposon Tagging and Gene Disruption. Nature, 1999. **402**(25): p. 413-418.

[7] Ewing, B. and P. Green, Base-calling of automated sequencer traces using phred II. Error probabilities. Genome Research, 1998. **8**: p. 186-194.

[8] Ewing, B., L. Hillier, M. Wendl, and P. Green, Base-calling of automated sequencer traces using phred I. Accuracy assessment. Genome Research, 1998. **8**: p. 175-185.

[9] Altschul, S., W. Gish, W. Miller, E. Myers, and D. Lipman, Basic Local Alignment Search Tool. Molecular Biology, 1990. **215**: p. 403-410.

[10] Worley, K., B. Wiese, and R. Smith, BEAUTY: an enhanced BLAST-based search tool that integrates multiple biological information resources into sequence similarity search results. Genome Res., 1995. **5**(2): p. 173-84.

[11] Cheung, K., A. Kumar, M. Snyder, and P. Miller, An Integrated Web Interface for Large-Scale Characterization of Sequence Data. Functional and Integrative Genomics, 2000. **1**: p. 70-75.

[12] Cheung, K., A. Deshpande, N. Tosches, S. Nath, A. Agrawal, P. Miller, A. Kumar, and M. Snyder. A Metadata Framework for Interoperating Heterogeneous Genome Data Using XML. AMIA 2001, in press.