Storing Efficiently Bioinformatics Workflows

Michel Kinsy and Zoé Lacroix
Arizona State University
PO Box 875706 Tempe AZ 85287-5706, USA

Abstract

We propose an efficient storage strategy to record bioinformatics *workflows. Our approach presents scientists with a flexible design model that distinguishes the scientific aim as a* design protocol *expressed against an ontology from the implementations(s), scientific workflows composed of bioinformatics services, and their execution. The storage strategy presented in this paper allows efficient access and constitutes the framework for reasoning on scientific protocols and experimental data.*

1 Introduction

The field of *bioinformatics* has grown immensely in the last 20 years; this growth is due to the intense research efforts invested in analyzing protein structures and sequencing of genes. Those efforts have produced vast amounts of data and raised different storage and query challenges not typically encountered with business data. Various works have been done to address the functionality and the capacity of scientific databases [7]. These databases and repositories typically do not record the processes producing the data they are storing, leaving scientists with no access to information relevant to data provenance needed to evaluate the quality of the results produced at execution. Moreover, complete protocol information including the description of the process and the selection of the resources used to implement it is critical to reproduce the experiment thus validates the scientific soundness of the results. In the recent years different software projects have been initiated to assist scientists to design, implement, and store their workflows and the data collected by the execution of these workflows [12, 11, 2]. But these systems in their attempts to be general scientific workflow design tools, have failed to provide protocol storage and to optimize the characteristics most needed and useful in bioinformatics workflow design, implementation, storage, and querying.

ProtocolDB is designed to provide scientists with a software tool allowing them to design and manage their workflows exploiting a domain consistent with the semantics of the experiments, store, and query both workflow design and implementation as well as the data collected at workflow execution. In ProtocolDB, scientific protocols record both the scientific aim of each task and the description of its implementation. To offer flexibility, we decompose each scientific protocol into two components: *design* and *implementation*. Both the design and the implementation of a scientific protocol are composed of coordinated tasks. Each task of the protocol design is defined by its input, output, and description. When an ontology is available to describe the scientific objects and tasks involved in a scientific protocol, the input and output of each protocol design task are defined by their respective concept classes. The description of the task may be a relationship defined between the input and output conceptual classes or a description of a relationship not defined in the ontology. The protocol implementation describes the selection of resources used to implement each task of the protocol. The description of an implementation task is the application, or service (e.g., Web service [1], BioMOBY [13]) selected to implement the corresponding design task. The input (respectively output) of a protocol implementation task is the description of the data collection input (respectively output) format of the service. Input data are instances of the input conceptual class of the corresponding design task. A similar distinction of the scientific aim from its implementation was noted in [5] with workflow conceptualization and specification, and in [3] with abstract and concrete workflows.

2 Motivating Example

Alternative Splicing (AS) is a splicing process of a pre-mRNA sequence transcribed from one gene that leads to different mature mRNA molecules thus to different functional proteins. Alternative splicing events are produced by different arrangements of the exons of a given gene. The Alternative Splicing Protocol (ASP) we present in this section is currently supporting the BioInformatics Pipeline Alternative Splicing Services

1-4244-1509-8/07/$25.00 ©2007 IEEE

BIPASS [9].

The Alternative Splicing Protocol (ASP) takes a set of transcripts as input and returns clusters of transcripts aligned to a gene. The process of alignment consists of an alignment of each transcript sequence against each genomic sequence of a whole genome of one or more organisms. This step is executed with all known transcripts extracted from different public databases. A clustering step immediately follows the alignment step. That step allows delimiting the transcript region of a gene excluding its regulation region. A cluster normally represents or may be representative of all intermediate transcripts (from the pre-messenger-RNA(s) to the mature messenger-RNA(s)) required to obtain one or several functional translated proteins from the same gene.

A single protocol *design step* such as the alignment step of ASP may be mapped to a complex *implementation protocol*. The motivations for such a complex implementation protocol are detailed and analyzed in [8]. The protocol implementation corresponding to the alignment task of ASP is composed of seven tasks illustrated in Figure 1.

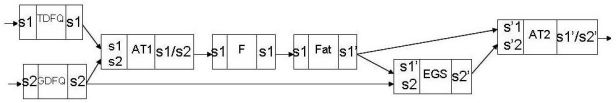

Figure 1. Initial Design Task Decomposition

First the input sequences are retrieved from local or public data sources. Transcript Data Filtering/Queries (**TAFQ**) and Genomic Data Filtering/Queries (**GAFQ**) denote queries against local or external databases to retrieve the input sequence query and the sequence database against which it is aligned. The input sequences are submitted to BLAT, an alignment tool that eliminates roughly all sequences not likely to produce a fine alignment with the input. **S1** and **S2** denote the Transcript and Genome data flows respectively, and BLAT is selected to implement **AT1**. Once the first alignment step is performed, the first 10% of the output are selected (**F**). Then the aligned transcripts are extracted from the previous steps (branch 1) and the aligned genomic sequences are extracted from the previous step (branch 2). **Fat** denotes the filtering function that removes all transcripts aligned but failed under a given threshold percentage value. **EGS** extracts the exact part of the matching genomic sequences and adds a defined number of bases in upstream and downstream of the extracted genomic sequence. The resulting transcript sequences **S1** and genomic sequences **S2** are submitted to a second alignment tool that performs a fine alignment step. **AT2** is implemented with SIM4 while **S1'/S2'** represents the final output of the whole alignment.

3 Protocol Model Analysis

Scientific workflows expressed in workflow system such as Taverna [12], Kepler [11], or SemanticBio [10] are typically driven by interoperability. They are composed of bioinformatics services that are connected sequentially or in parallel for execution. These systems do not offer a protocol analysis that could generate an equivalent protocol that could be more efficient for storage purposes, execution, or data provenance analysis. In this section we analyze scientific protocols and show that Petri nets offer a valuable model to represent and optimize scientific protocols.

An equivalent representation of the protocol shown in Figure 1 with a Finite State Machine (FSM) with a Source state and a Sink state (introduced for completeness) is shown in Figure 2. The FSM representation of the protocol may easily be converted into a Petri net by representing each state or edge of the FSM by a firing transition or process.

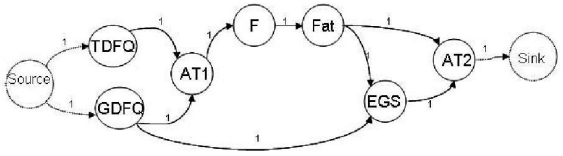

Figure 2. FSM of Design Task

The Petri net obtained from the transformation of the above automaton is shown in Figure 3. Such representation is an abstraction of the dataflow that captures the internal structure resulting from the sequential or parallel connectors used to define the protocol.

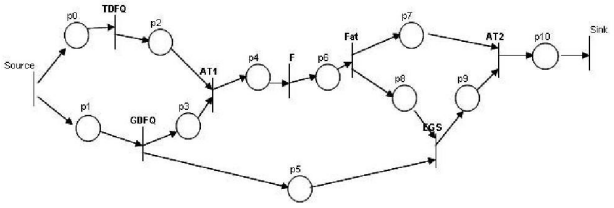

Figure 3. Petri Net of Design Task

The incidence matrix of the protocol is defined by $\mathbf{A} = (\mathbf{a}_{ij})$, where $\mathbf{a}_{ij} = \mathbf{e}(t_i, p_i) - \mathbf{e}(p_j, t_j)$, and where t_i is a transition (e.g., **AT1**), p_i is a process (e.g., **p4**), and $\mathbf{e}(n, m)$ is the number of incoming edges from n to m.

Example The incidence matrix A of the alignment protocol illustrated in Figure 2 is computed as follows. Each column corresponds to a process $p_0, \ldots, p_{10}$ and each raw corresponds to a workflow transition Source, **TDFQ**, **GDFQ**, **AT1**, **F**, **Fat**, **EGS**, **AT2**, and Sink. The incidence matrix obtained from the Petri net of the design task decomposition is shown in Figure 4.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

Figure 4. ASP incidence matrix

The Petri net simulates the workflow as follows. The initial state is represented by a vector $\vec{x}_0$ and each firing vector identifies the transition fired in the Petri net. Given a state $\vec{x}_i$ and a firing vector $\vec{v}_i$, $\vec{x}_{i+1} = \vec{x}_i + \vec{v}_i \mathbf{A}$.

Example Firing the transition **Fat** corresponds to the firing vector [0,0,0,0,0,1,0,0,0]. From the initial state $\vec{x}_0 = [0,0,0,0,0,0,0,0,0,0,0]$, the successive application of firing rules generates the covering tree of the workflow illustrated in Figure 5. The coverability tree is the set of all states of the system reachable from the initial state and thus represents all valid implementations that match the design task. There are various algorithms proposed to efficiently and automatically determine the covering tree [4, 6].

The approach allows the representation (thus the storage) of complex implementation workflows in the terms of a simple mathematical model which conserves semantics and use syntaxes that can be traced back to the original design. Furthermore, although the graph only represents data dependencies, the model is robust enough to support other types of dependencies. For example, in addition to the data dependencies we could associate with each task a time constraint as follows. For each task T, there is a couple (t_e, t_l) where t_e represents the earliest time the task can be executed and t_l is the latest time the task can be executed to preserve the over dependencies in the workflow. The time constraint capacity of the model allows us to not only optimize a design in terms of steps but also in term of scheduling for query planning. Other paramaters may be associated to tasks to capture various measures relevant to workflow execution. Such analysis so far has been greatly ignored by other bioinformatics workflow data

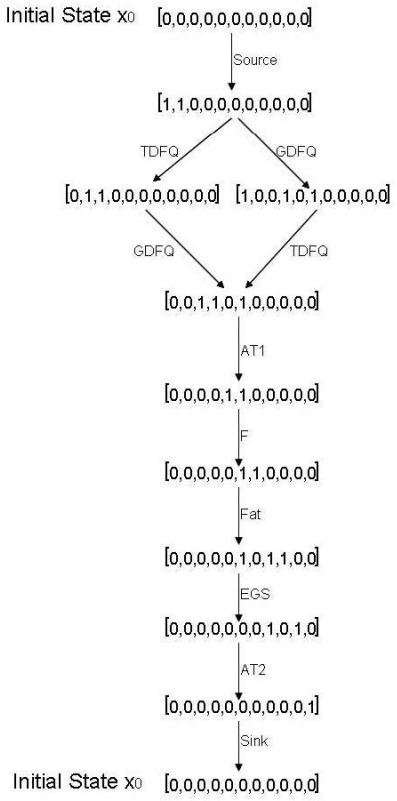

Figure 5. Coverability Tree of the Petri Net

analysis and storage approaches. It is crucial when dealing with these vast amounts of data and the sort of data intensive analyses over large distributed systems as encountered in bioinformatics to have a model that also addresses workflow performance.

4 Workflow Implementation Model

In our approach a workflow *implementation* is achieved in two steps: an *implementation specification* step where the workflow is expressed in terms of tangible, available resources: such as input and output format or data type, laboratory tools and applications; and a *design/implementation mapping*.

4.1 Implementation Specification

The *implementation protocol* that represents the executable workflow is consistent with the *design protocol* which captures the semantics of the workflow. If a given design task translates into a composition of tasks in the implementation model, a local refinement of the design task is created and attached to the specific implementation such that each implementation task is mapped directly to a design task. The fact

that the design is abstracted and represented by a matrix automatically promotes modularity of the design through a matrix partitioning[1] and fully supports re-decomposition at the implementation stage. Matrix partitioning is a mature technique for splitting a very large matrix into smaller, easier to store and more manageable sub-matrices for which key characteristics are easily determined. In this model as in any graph matrix representation, depending on the complexity and the size of the workflow, incidence matrix can be large and filled with zeros, therefore matrix partitioning allows conformity in storage, reduction in redundant data and block grouping for modularity.

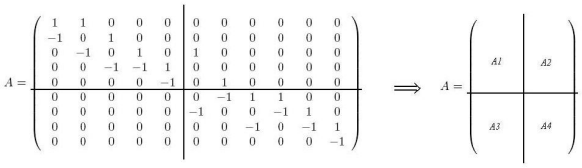

Figure 6. Matrix Partitioning

Example We partition the ASP incidence matrix (see Figure 6) and although the matrix partitioning was, in this example, arbitrary the resulting sub-matrices are still coherent sub-graphs of the Petri net illustrated in Figure 7, demonstrating the consistency of the design and the robustness and the degree of modularity of the model.

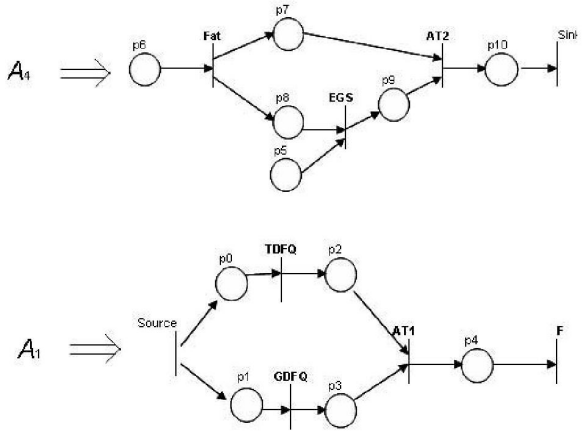

Figure 7. Sub-workflows

Such modeling supports add-ons and updates which can be easily performed without the overhead of reformatting the whole design graph. For example in our current illustration design we could replace the process $p4$ by some complex sub-graph without modifying the rest of the design and only be concerned with the consistency of the data representing the edge coming and leaving $p4$. It is worth noticing that to fully support and reinforce data dependencies or other constrains during decomposition, uniformity and uniqueness in the naming (IDs) of tasks must be adopted by the designer, but this requirement is not an additional overhead because the storage and the query of tasks already imposed such requirement on the overall workflow design. This technique also allows the system to search for certain patterns in the design and to update a given design by searching for the most common decomposition of a design task at the implementation stage.

[1] Experiments on Sparse Matrix Partitioning by S. Riyavongy. CERFACS Working Note WN/PA/03/32 CERFACS, 42 Avenue G. Coriolis, 31057 Toulouse Cedex, France.

Table 1. Graph Storage Table

Source	Destination	Weight of the Edge
Source	$TDFQ$	1
Source	$GDFQ$	1
$TDFQ$	$AT1$	1
$GDFQ$	$AT1$	1
$AT1$	F	1
F	$AT2$	1
Fat	EGS	1
Fat	$AT2$	1
EGS	$AT2$	1
$AT2$	*Sink*	1

4.2 Workflow Storage Schema

The storage of the design workflow is done at two different levels of abstraction, although any one of them fully expresses the design. First the graph is stored in a table *(source, destination, weight of edge)*. For any given two connected tasks in the workflow, the *source* attribute is the preceding task's name or ID, the *destination* attribute is the successor task of the source, and the *weight of edge* is the number of data edges from the source's output to the destination's input. For example the graph storage corresponding to Figure 2 is displayed in Table 1.

The graph table could be the only record of the workflow, but the computation and analysis of the Petri net and its matrix would have to be computed for each implementation of the workflow. To eliminate this calculation we propose to store a compressed version of its incidence matrix. This solution will avoid massive matrix data storage because. Indeed as the workflow gets largeer the matrix size increases, even with the matrix partitioning. The compression matrix storage has the following schema: *(ti, process pjm ,...,*

process p_{jn}, $w_m,\ldots,w_n$), where $w_m,\ldots,$ w_n are the numeric values of $a_{ti,}p_{jm}$.. $a_{ti,}p_{jn}$ found in the matrix, with $a_{ti,}p_{jx}$ non zero. For example, **A** will be represented with (Source, p0, p1, 1, 1), (TDFQ, p0, p2, -1, 1), ..., (sink, p10, -1) and recorded as follows.

Workflow Design	Id	Name	Creator	Date of Creation	Update	Person	Task_Table	Matrix_Table	Description
	01	Alternative Splicing (SA)	John Doe	05/12/2007 1:35PM	05/15/2007 4:12PM	By Uncle Sam	TT1	MT1	Alternative Splicing (AS) is a splicing process of a pre-mRNA sequence transcribed from one gene that leads to different mature mRNA molecules thus to different functional proteins. Alternative splicing events are produced by different arrangements of the exons of a given gene
	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	-

Matrix Data ID:MT1	Transition ID	Transition Name	Edge_Table ID
	01	Source	e1
	02	TDFQ	e2
	03	GDFQ	e3
	04	AT1	e4
	05	F	e5
	06	Fat	e6
	07	EGS	e7
	08	AT2	e8
	09	Sink	e9

Edge Data ID: e1	Process ID	Edge Weight
	p0	1
	p1	1

5 Conclusion and Future Work

ProtocolDB currently under development at Arizona State University is a system for scientific protocol management, including creating, storing, querying, and analyzing scientific workflows. The key components of the system consist of a friendly user interface to design and implement workflows, a robust relational database to store workflows and their associated data. In this paper we present preliminary results related to strategies for optimal protocol representation to optimize protocol access. Future work will focus on exploiting workflow equivalences to improve workflow execution, workflow storage, and reasoning on data provenance.

Acknowledgments This research was partially supported by the National Science Foundation[2] (grants IIS 0223042, IIS 0431174, IIS 0612273, and IIS 0738906). Michel Kinsy conducted the work while completing his undergraduate studies at ASU. The authors would like to thank Piotr Wlodarczyk and Christophe Legendre for their valuable input, and Natalia Kwasnikowska and Jan Van den Bussche for multiple discussions on scientific workflows.

References

[1] David Booth and Canyang Kevin Liu. Web Services Description Language (WSDL) Version 2.0 Part 0: Primer. W3C Working Draft, December 2004. http://www.w3.org/TR/2004/WD-wsdl20-primer-20041221/.

[2] Any opinion, finding, and conclusion or recommendation expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

[2] David Churches, Gabor Gombas, Andrew Harrison, Jason Maassen, Craig Robinson, Matthew Shields, Ian Taylor, and Ian Wang. Programming Scientific and Distributed Workflow with Triana Services. *International Journal on Concurrency and Computation: Practice and Experience*, 18(10):1021–1037, 2006.

[3] Ewa Deelman, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Sonal Patil, Mei-Hui Su, Karan Vahi, and Miron Livny. Pegasus: Mapping Scientific Workflows onto the Grid. In *European Across Grids Conference*, pages 11–20, 2004.

[4] Alain Finkel. Decidability of the termination problem for completely specified protocols. *Distributed Computing*, 7(3):129–135, 1994.

[5] Nada Hashmi, Sung Lee, and Michael P. Cummings. Abstracting workflows: unifying bioinformatics task conceptualization and specification through semantic web services. In *W3C Workshop on Semantic Web for Life Sciences*, Cambridge, Massachusetts, USA, 2004.

[6] Didier Cristani Alain Jean-Marie and Christine Coves. Petri net analysis: Complexity and finite coverability graph in modular design. *Studies in Informatics and Control*, 14(1):54–64, 2005.

[7] Zoé Lacroix and Terence Critchlow, editors. *Bioinformatics: Managing Scientific Data*. Morgan Kaufmann Publishing, 2003.

[8] Zoé Lacroix and Christophe Legendre. Analysis of a Scientific Protocol: Selecting Suitable Resources. In *First IEEE International Workshop on Service Oriented Technologies for Biological Databases and Tools, In conjunction with ICWS/SCC*, pages 130–137, 2007.

[9] Zoé Lacroix, Christophe Legendre, Louiqa Raschid, and Ben Snyder. BIPASS: BioInformatics Pipelines Alternative Splicing Services. *Nucleic Acids Research*, Web Server Issue: W292–6, July 2007.

[10] Zoé Lacroix and Hervé Ménager. SemanticBio: Building Conceptual Scientific Workflows over Web Services. In *Data Integration in the Life Sciences*, volume 3615 of *Lecture Notes in Computer Science*, pages 296–299. Springer, 2005.

[11] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific Workflow Management and the KEPLER System. *Concurrency and Computation: Practice and Experience, Special Issue on Scientific Workflows*, 18(10):1039–1065, 2005.

[12] Thomas M. Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, R. Mark Greenwood, Tim Carver, Kevin Glover, Matthew R. Pocock, Anil Wipat, and Peter Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.

[13] Mark D. Wilkinson and Matthew Links. BioMOBY: an open-source biological web services proposal. *Briefings in Bioinformatics*, 3(4):331–341, December 2002.