Genetics and Molecular Biology, 27, 4, 691-695 (2004)
Copyright by the Brazilian Society of Genetics. Printed in Brazil
www.sbg.org.br

Research Article

GenFlow: Generic flow for integration, management and analysis of molecular biology data

Marcio Katsumi Oikawa[1], Marcos Eduardo Bolelli Broinizi[1], Alexandre Dermargos[2], Hugo Aguirre Armelin[2] and João Eduardo Ferreira[1]

[1]*Universidade de São Paulo, Instituto de Matemática e Estatística, Departamento de Ciência da Computação, São Paulo, SP, Brazil.*
[2]*Universidade de São Paulo, Instituto de Química, Departamento de Bioquímica, São Carlos, SP, Brazil.*

Abstract

A large number of DNA sequencing projects all over the world have yielded a fantastic amount of data, whose analysis is, currently, a big challenge for computational biology. The limiting step in this task is the integration of large volumes of data stored in highly heterogeneous repositories of genomic and cDNA sequences, as well as gene expression results. Solving this problem requires automated analytical tools to optimize operations and efficiently generate knowledge. This paper presents an generic flow model , called GenFlow, that can tackle this analytical task.

Key words: biological workflow, information modeling, semantic and control data, molecular sequence analysis.

Received: November 11, 2003; Accepted: October 7, 2004.

Introduction

Computational molecular biology uses computational power to analyze genetic, molecular and biochemical data from multiple biological species in a genome scale. Presently, scientists face the challenge of integrating biological data stored over the years in dynamic heterogeneous structures. Many genomic and molecular biology studies have been applying a number of computational tools, generating results with different input and/or output formats, without common standards. Today, integration and analysis of these large volumes of heterogeneous data are no longer amenable to direct human manipulation and has become one of the major problems in bioinformatics. Thus, biologists badly need efficient dynamic techniques for data integration. However, dynamic integration and analysis of heterogeneous data, using multiple applications (applying different heuristics on the data, customizing specific arguments, etc.), are not trivial problems. In fact, they remain important topics for research in computer science.

This paper reports on the development of an generic flow model, GenFlow, allowing efficient integration and analysis of heterogeneous biological data. We start characterizing the generic flow approach describing the use of GenFlow to tackle a common problem in molecular genetics, *i.e.*, choosing molecular clones from a mouse cDNA library to manufacture microarrays for studies of differential gene expression during cell cycle progression. Next, we present a formal study of generic flow design. The final product, GenFlow, provides integrated tools enabling investigators to comparatively analyze results of multiple experiments, through friendly structures for management of heterogeneous data.

Sendo correspondence to Joao Eduardo Ferreira. Universidade de São Paulo, Instituto de Matemática e Estatística, Departamento de Ciência da Computação, Rua do Matao 1010, 05508-090 São Paulo, SP, Brazil. E-mail: jef@ime.usp.br.

Characterizing the Problem: A Common Work Flow in Molecular Genetics

We illustrate the integration problem and the generic flow approach with a real example found in studies of mouse gene expression within a large cooperative project between departments of the University of São Paulo (CAGE-Cooperation for Analysis of Gene Expression). We had an UNIGENE set of 34,000 mouse cDNA clones (kindly supplied by Prof. Marcelo Bento Soares, University of Iowa; Bonaldo *et al.*, 1996) whose sequences stored in the Genbank database were the only data available. To select clones of interest, we proceed according to the following steps:

1. Assembly of the sequences, performed by Phrap (Gordon, 2001) and CAP3 (Huang, 1996; Huang and Madan, 1999), aiming to confirm sequences uniqueness.

2. Identification and separation of polyA tail-possessing sequences.

3. Local alignment against a general database NR (NCBI, 1988), performed by BLAST (Altschul *et al.*, 1997), aiming to assess sequence similarities and to exclude no match sequences. At the end of this step, we were reduced to 25,772 sequences.

4. Another round of the Blast against a RIKEN (RIKEN, 2001) database, searching for similar sequences in the Mouse Consortium Genome. At this point we were down to 17,338 sequences.

5. Choice of the best sequences of interest, manually done by the investigator.

Basically, the generic flow is represented by the scheme in Figure 1.

Note that each application has a specific role, therefore, Phred, CAP3 and BLAST must be executed according to the specified order. Thus, the generic flow is a single sequential chain, with well-defined starting and finishing points and without loops between steps (see Figure 1).

Formal Analysis and generic flow Modeling

GenFlow (Generic flow) stores the results of generic components for pipeline definition and control, seeing the applications as atomic objects. These objects encapsulate some properties while maintaining mutual relationships. These relationships, adding to the knowledge of the expert users, define a flow for information. The Genflow can be edited for supporting more than one application and generalized for multiple users. Particularly, we have presented an example applied to sequence analysis, but it could be used to analyze microarray data, metabolic and signaling pathways, protein 3D structures, and so forth.

Next, we will suggest a formal way of modeling a generic execution flow, built from the applications and based on the following assumptions:

i. there is a finite and non-empty collection of applications for analysis, which we consider valid for the operating system. This is denoted as A;

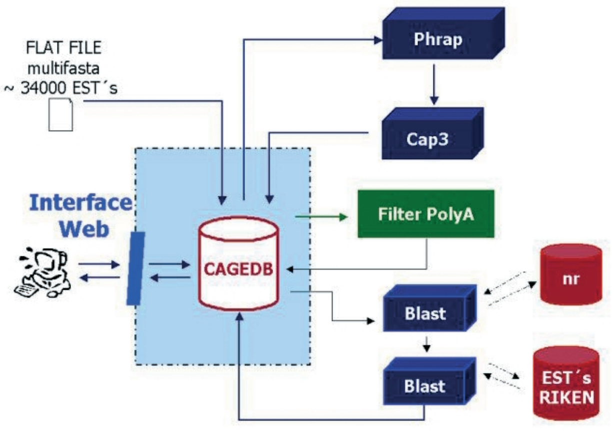


Figure 1 - Pipeline for Gene Classification.

ii. there is a finite set of non-empty values for tasks, denoted as T. This is a set of discrete elements, where each element indicates the position of the application within the workflow;

iii. there is a finite set of finite and non-empty values for the algorithms, denoted as H. The algorithms indicate the variation of the implementation strategies for the execution of the same task;

iv. there is a finite set of finite and non-empty values for structural input and output formats, denoted as F;

v. some constructs, such as *tuple*, used for complex data structures composition, are available.

All values of the sets defined above (i to v) are atomic. Values are formed using the construct *tuple* and they will be called structured values (Liu, 1992).The type T denotes a task of the application p, $p \in A$, in the experiment.

Structure for applications

Application for analysis

Given a set P of parameters for an application, we can define an application for analysis as a *tuple*

$A = (v, h, t, p, in, out)$

where $v \in N$ and indicates the version of the used application,

$h \in H$, $t \in T$,

$p \in P$, $in \in F$

and $out \in F$.

Then, the set of all application objects are represented by $N \times H \times T \times P \times F \times F$. We will adopt the convention *object domain* in recovering the values of a particular domain.

Property Functions

We can use the above definition for relationships among applications. We call these relationships validators, which map their arguments to Boolean values (true or false).

Precedence validator (<)

The $<$ is a function of the form f: $A \times A \rightarrow \{\text{true, false}\}$.

Let a and b be applications, whereby a, $b \in A$ and $a \neq b$. We say that $a < b$ (or a precedes b) if a and b have distinct types and the type of a ($a.t$) indicates a task prior to type of b ($b.t$);

Format equivalence validator (≈)

The $\approx$ is a function of the form f: $F \times F \rightarrow \{true, false\}$.

Let a and b be structured formats, whereby a, $b \in F$. We say that $a \approx b$ if a has an equal semantic meaning to b, and, it is so possible structurally to translate a to b.

Moreover, *a* and *b* must belong to the same domain. In addition, each structure component of an *a* format must have an equivalence to *b* format. A direct consequence of this approach is the usage of a building wrapper to translate files from *a* to *b* formats.

Chained validator ($\rightarrow$)

The $\rightarrow$ is a function of the form f: $A \times A \rightarrow \{true, false\}$.

Let *a* and *b* be applications, whereby $a, b \in A$. We say that $a \rightarrow b$ (or *a* chains *b*) if $a < b$, $a.t = b.t$ and $a.out \approx b.in$.

Therefore, *a* and *b* are called chains if we can run them in series, using the output of application *a* as input (or one of them) in application *b*, abstracting structural format restrictions for input/output files. Hence, *a* generates an output file semantically similar to the *b* input file.

Application equivalence validator ($\equiv$)

The $\equiv$ is a function f: $A \times A \rightarrow \{true, false\}$.

Given applications *a* and *b*. We assume that $a, b \in A$. We denote by $a \equiv b$ (or *a* is equivalent to *b*) whether *a* and *b* are of the same type ($a.t = b.t$) and, in addition, whether their input and output files are semantically similar; this implies that $a.in \approx b.in$ and $a.out \approx b.out$.

Relationships and communication

We can define some logical relations and their communication features using the definitions viewed previously. In this way, we can classify the applications according to their relations and, in addition, we are able to construct communication rules and restrictions based on their properties.

Chained applications

Let P_i and P_j be two applications, whereby $P_i, P_j \in A$. P_i and P_j are called chained if the relation $P_i \rightarrow P_j$ returns *true*.

The applications chain is a relation key for our integration system. This relationship defines whether they can be executed in sequential order, or not. Furthermore, this relation is strongly linked to the input/output structures of the used algorithms, because these patterns define, physically, the sets of applications which can be chained.

Modeling and implementation

Biological data processing usually generates a sequential flow of tasks, each one representing a single application. The full, or partial automation, occurs through an integration process, controlling communication among subsequent applications. Our approach extends ideas from Peleg *et al.* (2002) and Siepel *et al.* (2001), integrating tasks according to running rules specified and formalized earlier in this section.

The precedence and chaining logic, represented by respective functions and exposed by their definitions, allow us to create alternative flows for the experiment analysis. The formalization facilitates the design of generic integration components, allowing better definition of flow structures. In addition, formal components give us the freedom to build complex generic flow, analyzing them through diverse viewpoints. GenFlow allows partial flow without having to cover all sequential tasks. This is very convenient, since there are cases in which we do not have primitive data sources or our experiment tasks are based on intermediate results.

The main advantage of a GenFlow editing system is the automation of sequencing tasks. A file generated from one step can be used in next, even if it has some structural incompatibility, solved by a XML file translation (Achard *et al.*, 2001) (see the internal structure of the system in Figure 2 and the interface for the users in Figures 3 and 4).

In addition, the real physical location of each application is visualized by users. Therefore, investigators do not worry about the installation or communication between ap-

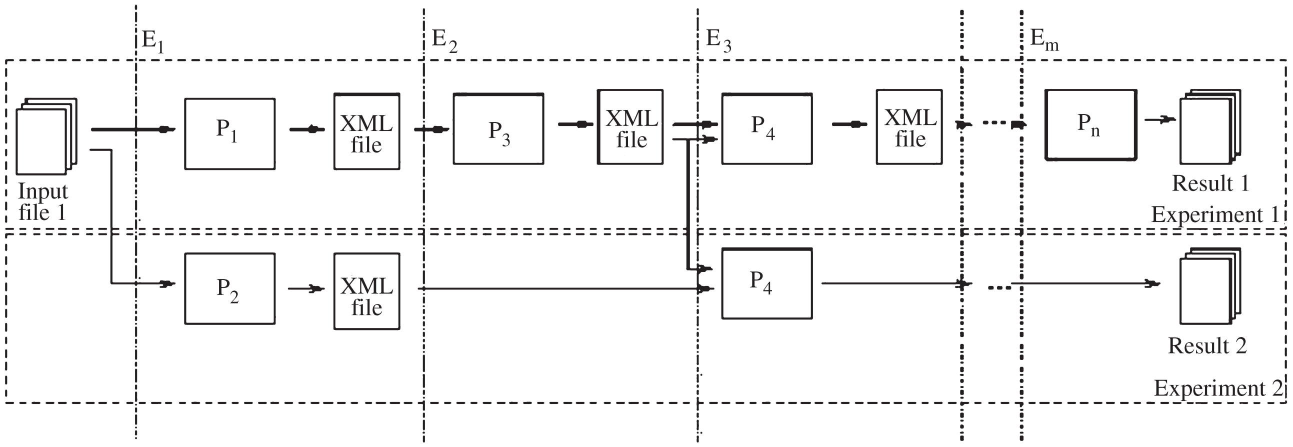
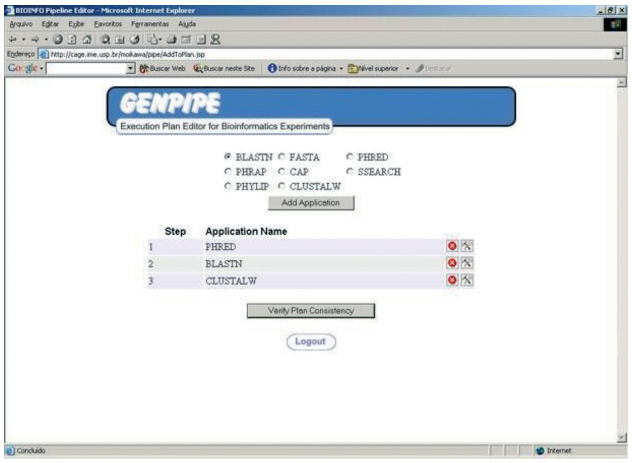
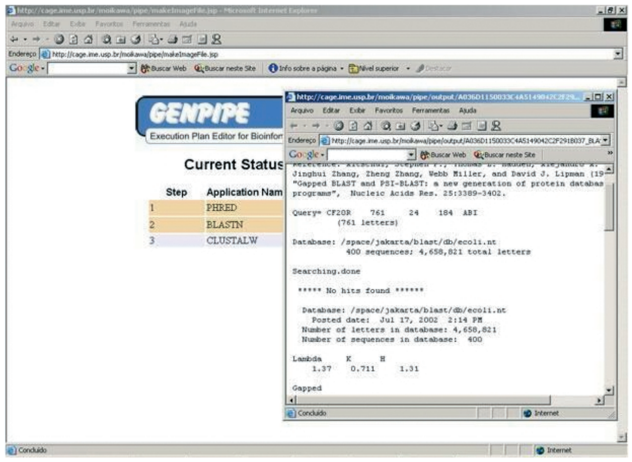


Figure 2 - GenFlow schema for molecular biology experiments.


BIOINFO Pipeline Editor - Microsoft Internet Explorer
http://cage.ime.usp.br/moikawa/pipe/AddToPlan.jsp
GENPIPE
Execution Plan Editor for Bioinformatics Experiments
BLASTN
FASTA
PHRED
PHRAP
CAP
SSEARCH
PHYLIP
CLUSTALW
Add Application
Verify Plan Consistency
Logout

Step	Application Name
1	PHRED
2	BLASTN
3	CLUSTALW

Figure 3 - Definition of the GenFlow: editing the task flow for an experiment.


http://cage.ime.usp.br/moikawa/pipe/makeImageFile.jsp - Microsoft Internet Explorer
GENPIPE
Execution Plan Editor for Bioinfor
Current Status
http://cage.ime.usp.br/moikawa/pipe/output/A036D1150033C4A5149042C2F29...

Step	Application Nam
1	PHRED
2	BLASTN
3	CLUSTALW

```
Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (19
"Gapped BLAST and PSI-BLAST: a new generation of protein databas
programs", Nucleic Acids Res. 25:3389-3402.

Query= CF20R    761    24    184  ABI
         (761 letters)

Database: /space/jakarta/blast/db/ecoli.nt
           400 sequences; 4,658,821 total letters

Searching.done

 ***** No hits found ******

  Database: /space/jakarta/blast/db/ecoli.nt
    Posted date:  Jul 17, 2002  2:14 PM
  Number of letters in database: 4,658,821
  Number of sequences in database:  400

Lambda     K      H
    1.37   0.711    1.31

Gapped
```

Figure 4 - Running GenFlow.

plications, because software interactions are controlled by GenFlow.

A generic GenFlow schema can be viewed in Figure 2. The applications P_i, i = 1,2, ... n (where *n* is the number of applications installed and available in the operating system), are placed in sequential order, according to their *task*. This property is defined when P_i is installed in the environment, according to its algorithm running parameters. It defines precedence relations among other applications.

Each P_i application is associated to an execution step E_k, k = 1,2, ..., m (where *m* is the number of tasks for a particular workflow). The chains are built in series and run using input/output files.

All applications follow the precedence order and present semantic compatibility of their input and output files; they are called chained. In Figure 2, we can see some kinds of chaining applications, such as P_1 and P_3, P_2 and P_4, P_3 and P_4.

The system implementation is external, using text files and database connection drivers. This approach produces a flexible architecture, under DBMS (Database Management System) and implementation language viewpoint. Between each two sequential steps, one or more text files are generated for intermediation of the communication interfaces of the applications; we usually map its information to a XML (W3C, 2004) file format. The choice of XML is based on the adoption of a renowned and standardized file format, optimizing the understanding of all embedded structural elements and promoting the improvements and extensions for interested groups. Furthermore, this preference avoids the creation of new, non-standard and eventually confusing file formats. The information in XML format can be stored in database through persistent services.

System Application and Conclusions

Our group is developing a web application of GenFlow. A centralized architecture was chosen for the web implementation, benefiting investigators that have facilities to manage applications and large volumes of data. The system can be accessed from different web clients, providing an additional level of convenience and freedom to the users.

In addition, all information is stored on a DBMS, guaranteeing information persistence and an alternative tool for data organization and retrieval, essential for works involving large amounts of data.

The visualization pages will allow multiple ways of displaying results, including construction of novel and customized forms to exhibit results.

Currently, integration research addresses the problem through scripts or structured compiled code, so each minimal change implies code modification. The main advantage of our approach is system scalability. We can reduce the dependence levels through software components, providing flexibility and dynamism for the system.

The dynamism is particularly interesting to biologists, because they will be able to start the execution of many workflows at the same time, without having to translate formats between the applications. Furthermore, the researcher gains evaluation power for algorithms, results and experiments, thus improving their quality.

In the Figures 3 and 4, we see the operation of the system. This process is divided into four steps:

1. Editing - the user builds the sequence of tasks (GenFlow) that need to be executed (see Figure 3);

2. Parameter configurations - the user sets up the parameters for each selected task, if necessary;

3. Data recovery - the user chooses the data that will feed GenFlow from a central DBMS;

4. Execution - the execution of the tasks starts (see Figure 4).

The addition of new applications into the system does not require strategic changes, because the integration level is separated from the applications. We need just define the function mapping to link the XML standard format of the system to the input and output formats of the new entry.

We successfully tested the performance of the GenFlow system analyzing a mouse cDNA library, presented in section 2. The investigators were able to integrate biological knowledge and automated execution of applications, without manual manipulations of the intermediate files.

Each step of this testing operation can be compared to a single filter. The filter determines the better information and aggregates quality gauges on the data. The combination of all filters comprising the whole system qualifies GenFlow as an generic flow system.

Acknowledgments

This paper is part of the project entitled "Integration of Data and Applications for computational molecular biology", funded by the Research Support Foundation of São Paulo State (FAPESP), grant number 00/10062-3. H.A.A. and A.D.O. are independently supported by FAPESP and CNPq.

References

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

Altintas I, Bhagwanani S, Buttler D, Chandra S, Cheng Z, Coleman M, Critchlow T, Gupta A, Han W, Liu L, Ludäscher B, Pu C, Moore R, Shoshani A and Vouk MA (2003) a modeling and execution environment for distributed scientific workflows. In: International Conference on Scientific and Statistical Database Management pp 247-250.

Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W and Lipman DJ (1997) Gapped blast and psi-blast: A new generation of protein database search programs. Nucleic Acids Research 25:3389-3402.

Bonaldo MF, Lennon G and Soares MB (1996) Normalization and subtraction: Two approaches to facilitate gene discovery. Genome 6:791-806.

Ellis GR (1995) Managing complex objects. PhD Thesis, University of Queensland, Australia.

Gordon D, Desmarais C and Green P (2001) Automated finishing with autofinish. Genome Research 11:614-625.

Gouet P and Courcelle E (2002) ENDscript: A workflow to display sequence and structure information. Bioinformatics 18:767-768.

Huang X (1996) A improved sequence assembly program. Genomics 33:21-31.

Huang X and Madan A (1999) CAP3: A DNa sequence assembly program. Genome Research 9:868-877.

Hubbard T (2002) Biological information: Making it accessible and integrated (and trying to make sense of it). Bioinformatics 18(90002):S140.

Koster R, Black AP, Huang J, Walpole J and Pu C (2001) Infopipes for composing distributed information flows. ACM International Workshop on Multimedia Middleware.

Liu L (1992) A formal approach to structure, algebra and communication of complex objects. PhD Thesis, Proefschrift Katholieke Universiteit Brabant Tilburg, The Netherlands.

Paton NW, Khan SA, Hayes A, Moussouni F, Brass A, Eilbeck K, Goble CA, Hubbard SJ and Oliver SG (2000) Conceptual modelling of genomic information. Bioinformatics 16:548-557.

Payton J, Gamble R, Kimsen S and Davis L (2000) The opportunity for formal models of integration. International Conference on Information Reuse and Integration, Hawaii, USA.

Peleg M, Yeh I and Altman RB (2002). Modelling biological processes using workflow and Petri Net models. Bioinformatics 18:825-837.

Sheth A, Georgakopoulos D, Joosten SMM, Rusinkiewicz M, Scacchi W, Wileden J and Wolf A (1996) State-of the-art and future directions. NSF Workshop on Workflow and Process Automation in Information System, Atlanta, USA. http://lsdis.cs.uga.edu/activities/NSF-workflow.

Siepel A, Farmer A, Tolopko A, Zhuang M, Mendes P, Beavis W and Sobral B (2001) ISYS: A decentralized, component-based approach to the integration of heterogeneous bioinformatics resources. Bioinformatics 17:83-94.

Singh MP and Huhns MN (1994) Automating workflows for service order processing: Integrating AI and database technologies. IEEE Expert 9:19-23.

W3C-World Wide Web Consortium (2004) Extensible Markup Language (XML) Specifications and translations. http://www.w3.org/XML. Last visit on Sep 27, 2004.

National Center for Biotechnology Information - NCBI (1988) Entrez help document. http://www.ncbi.nlm.nih.gov/entrez/query/static/help/helpdoc.html. Last visit on Sep 27, 2004.

The RIKEN Genome Exploration Research Group Phase II Team and the FANTOM Consortium (2001), Functional annotation of 21,076 sequenced mouse cDNAs prepared from full-length enriched libraries. Nature 409:685-690.