

Norman W. Paton
Paolo Missier
Cornelia Hedeler (Eds.)

LNBI 5647

Data Integration in the Life Sciences

6th International Workshop, DILS 2009
Manchester, UK, July 2009
Proceedings

 Springer

Lecture Notes in Bioinformatics

5647

Edited by S. Istrail, P. Pevzner, and M. Waterman

Editorial Board: A. Apostolico S. Brunak M. Gelfand
T. Lengauer S. Miyano G. Myers M.-F. Sagot D. Sankoff
R. Shamir T. Speed M. Vingron W. Wong

Subseries of Lecture Notes in Computer Science

Norman W. Paton Paolo Missier
Cornelia Hedeler (Eds.)

Data Integration in the Life Sciences

6th International Workshop, DILS 2009
Manchester, UK, July 20-22, 2009
Proceedings

Series Editors

Sorin Istrail, Brown University, Providence, RI, USA
Pavel Pevzner, University of California, San Diego, CA, USA
Michael Waterman, University of Southern California, Los Angeles, CA, USA

Volume Editors

Norman W. Paton
Paolo Missier
Cornelia Hedeler
University of Manchester
School of Computer Science
Oxford Street, Manchester, M13 9PL, UK

E-mail:

npaton@manchester.ac.uk
pmissier@acm.org
chedeler@cs.manchester.ac.uk

Library of Congress Control Number: 2009929939

CR Subject Classification (1998): H.2.8, J.3, E.2, H.2, H.3.5

LNCS Sublibrary: SL 8 – Bioinformatics

ISSN 0302-9743
ISBN-10 3-642-02878-0 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-02878-6 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

springer.com

© Springer-Verlag Berlin Heidelberg 2009
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12706389 06/3180 5 4 3 2 1 0

Preface

Data integration in the life sciences continues to be important but challenging. The ongoing development of new experimental methods gives rise to an increasingly wide range of data sets, which in turn must be combined to allow more integrative views of biological systems. Indeed, the growing prominence of systems biology, where mathematical models characterize behaviors observed in experiments of different types, emphasizes the importance of data integration to the life sciences. In this context, the representation of models of biological behavior as data in turn gives rise to challenges relating to provenance, data quality, annotation, etc., all of which are associated with significant research activities within computer science.

The Data Integration in the Life Sciences (DILS) Workshop Series brings together data and knowledge management researchers from the computer science research community with bioinformaticians and computational biologists, to improve the understanding of how emerging data integration techniques can address requirements identified in the life sciences.

DILS 2009 was the sixth workshop in the series, and the associated proceedings contains 15 peer-reviewed papers and 2 invited papers. The papers explore a range of topics somewhat off the beaten track of classic data integration architectures. Sessions addressed: *graph-based modelling and integration*, in which graph-based models underpin analyses without costly up-front schema integration; *annotation*, reflecting the fact that much practical integration involves associating experimental or derived results with auxiliary information; *structure inference*, which seeks to automate the identification of structured models or their relationships; and *data and work flows*, whereby data and analyses are combined for biological data integration. These technical sessions were complemented by a session on *data integration for systems biology*, which brings a variety of techniques to bear on the management and integration of biological models. Invited presentations by Zachary Ives and Nicolas Le Novère, respectively, explored techniques for managing our evolving understanding of shared scientific data, and experience in modelling, annotating and sharing of systems biology models. As such, the workshop brought together results on a collection of different strands of data integration research, in particular reflecting the evolving nature of biological data sources and integration requirements.

The editors would like to thank the Program Committee and the external reviewers for their work in enabling the timely selection of papers for inclusion in the proceedings. We are also pleased to acknowledge the sponsorship of the ENFIN European Network of Excellence in Systems Biology

(<http://www.enfin.org/>), which also organized a colocated workshop. Finally, we are also grateful for the cooperation of Springer in putting this volume together.

July 2009

Norman Paton
Paolo Missier
Cornelia Hedeler

Organization

Executive Committee

General Chair	Paolo Missier	The University of Manchester, UK
Program Chair	Norman W. Paton	The University of Manchester, UK
Publicity	Suzanne M. Embury	The University of Manchester, UK
Proceedings	Cornelia Hedeler	The University of Manchester, UK

Program Committee

Shawn Bowers	University of California, Davis, USA
Sarah Cohen-Boulakia	LRI, Université Paris-Sud 11, France
Terence Critchlow	Pacific Northwest National Laboratory, USA
Christine Froidevaux	LRI, Université Paris-Sud 11, France
Carole Goble	The University of Manchester, UK
Graham Kemp	Chalmers University of Technology, Sweden
Jessie Kennedy	Napier University, UK
Zoe Lacroix	Arizona State University, USA
Patrick Lambrix	Linköpings University, Sweden
Ulf Leser	Humboldt University, Germany
Mong Li Lee	National University of Singapore, Singapore
Frédérique Lisacek	Swiss Institute of Bioinformatics, Switzerland
Bertram Ludäscher	University of California, Davis, USA
Marco Masseroli	Politecnico di Milano, Italy
Robin McEntire	GlaxoSmithKline, USA
Paolo Missier	The University of Manchester, UK
See-Kiong Ng	Institute for Infocomm Research, Singapore
José Luís Oliveira	University of Aveiro, Portugal
Oscar Pastor Lopez	Universidad Politecnica de Valencia, Spain
Norman Paton	The University of Manchester, UK
Alexandra Poulouvassilis	Birkbeck College, University of London, UK
Erhard Rahm	University of Leipzig, Germany
Louiqa Raschid	University of Maryland, USA
Falk Schreiber	IPK Gatersleben and MLU Halle-Wittenberg, Germany
Christian Stoeckert	University of Pennsylvania, USA
Thodoros Topaloglou	University of Toronto, Canada
Anil Wipat	Newcastle University, UK

Additional Reviewers

Alain Denise	LRI, Université Paris-Sud 11, France
Anika Gross	University of Leipzig, Germany
Michael Hartung	University of Leipzig, Germany
Cornelia Hedeler	The University of Manchester, UK
Markus Kirchberg	Institute for Infocomm Research, Singapore
Toralf Kirsten	University of Leipzig, Germany
Wee Siong Ng	Institute for Infocomm Research, Singapore
Daniel Zinn	University of California, Davis, USA

Sponsoring Institutions

ENFIN Enabling Systems Biology <http://www.enfin.org/>

Local Organization

Iain Hart	The University of Manchester, UK
Lynn Howarth	The University of Manchester, UK
Ruth Maddocks	The University of Manchester, UK
Lisa Ogden	The University of Manchester, UK
Bryony Quick	The University of Manchester, UK
Rina Sraibonian	The University of Manchester, UK

Website

DILS 2009 website <http://www.cs.manchester.ac.uk/DILS09/>

Table of Contents

Keynote Presentations

- Data Integration and Exchange for Scientific Collaboration 1
Zachary G. Ives
- Data Integration and Semantic Enrichment of Systems Biology Models
and Simulations 5
*Vijayalakshmi Chelliah, Lukas Endler, Nick Judy, Camille Laibe,
Chen Li, Nicolas Rodriguez, and Nicolas Le Novère*

Graph-Based Modelling and Integration

- Linking Life Sciences Data Using Graph-Based Mapping 16
*Jan Taubert, Matthew Hindle, Artem Lysenko, Jochen Weile,
Jacob Köhler, and Christopher J. Rawlings*
- Integration of Full-Coverage Probabilistic Functional Networks with
Relevance to Specific Biological Processes 31
Katherine James, Anil Wipat, and Jennifer Hallinan
- OpenFlyData: The Way to Go for Biological Data Integration 47
Jun Zhao, Alistair Miles, Graham Klyne, and David Shotton

Annotation

- On the Reachability of Trustworthy Information from Integrated
Exploratory Biological Queries 55
Eithon Cadag, Peter Tarczy-Hornoch, and Peter J. Myler
- Estimating the Quality of Ontology-Based Annotations by Considering
Evolutionary Changes 71
Anika Gross, Michael Hartung, Toralf Kirsten, and Erhard Rahm
- Integration and Mining of Genomic Annotations: Experiences and
Perspectives in GFINDER Data Warehousing 88
Marco Masseroli, Stefano Ceri, and Alessandro Campi

Structure Inference

- Site-Wide Wrapper Induction for Life Science Deep Web Databases 96
Saqib Mir, Steffen Staab, and Isabel Rojas

An Adaptive Combination of Matchers: Application to the Mapping of Biological Ontologies for Genome Annotation 113
Bastien Rance, Jean-François Gibrat, and Christine Froidevaux

Slicing through the Scientific Literature 127
Christopher J.O. Baker, Patrick Lambrix, Jonas Laurila Bergman, Rajaraman Kanagasabai, and Wee Tiong Ang

Data and Work Flows

Exploiting Parallelism to Accelerate Keyword Search on Deep-Web Sources 141
Tantan Liu, Fan Wang, and Gagan Agrawal

A Visual Interface for on-the-fly Biological Database Integration and Workflow Design Using VizBuilder 157
Shahriyar Hossain and Hasan Jamil

EpiC: A Resource for Integrating Information and Analyses to Enable Selection of Epitopes for Antibody Based Experiments 173
Niall Haslam and Toby Gibson

Data Integration for Systems Biology

Design and Architecture of Web Services for Simulation of Biochemical Systems 182
Joseph O. Dada and Pedro Mendes

An Integration and Analysis Pipeline for Systems Biology in Crop Plant Metabolism 196
Stephan Weise, Christian Colmsee, Eva Grafahrend-Belau, Björn Junker, Christian Klukas, Matthias Lange, Uwe Scholz, and Falk Schreiber

Towards Enhanced Retrieval of Biological Models through Annotation-Based Ranking 204
Dagmar Köhn, Carsten Maus, Ron Henkel, and Martin Kolbe

Author Index 221

Data Integration and Exchange for Scientific Collaboration

Zachary G. Ives

University of Pennsylvania, Philadelphia, PA 19104, USA
zives@cis.upenn.edu
<http://www.cis.upenn.edu/~zives>

1 Introduction

As the sciences have become increasingly data driven, it is clear that information integration is critical to their advancement. By integrating diverse data, we can allow biologists to discover big-picture patterns or behaviors, or to do comparative analyses among different organisms or systems. By enabling *collaborative editing and annotation* of integrated data — incorporating contributions from parties with different viewpoints — we can facilitate higher-quality, better-understood data. One of the open challenges, however, lies in developing the right *architectures and models* for supporting effective data integration and exchange in science.

The majority of work in the data integration literature was shaped by the needs of the enterprise and Web portals. Here, the application domain is often finite and well-understood, making it possible to develop a single standard schema; moreover, it is feasible for central management to mandate the adoption of such a standard. Data conflicts will generally arise only due to outdated information or dirty data. And finally, most data is fairly static. As a result, conventional data integration architectures are based on the notion of creating a single integrated schema, mapping all data into this schema, and applying data cleaning at the end to produce a single unified, consistent view of data.

Scientific data and its use are very different, largely because scientific understanding is constantly in flux — with a diversity of hypotheses, communities with different conventions, and specializations. Much of scientific data is not raw observations, but rather *hypotheses* based on these observations: e.g., an assertion of a relationship between two species or between a gene sequence and a morphological characteristic. Such hypotheses are often uncertain and the subject of dispute. They are constantly being revised as further experiments are conducted, or our understanding is otherwise improved.

Clearly, an enterprise-oriented view of data integration is mismatched for the needs of the life sciences. We instead need a data sharing scheme that:

- Accommodates multiple, community-specific schemas and vocabularies that may evolve over time.
- Supports highly dynamic, repeatedly revised, frequently annotated data.
- Facilitates sharing across different communities in a way that *scales with the amount of invested effort*: limited data sharing should be easy, and further time investment should enable greater data sharing.

- Tolerates disagreement among different communities about data items (hypothesized facts).
- Restricts the exchange of data based on assessments of *source authority* and *mapping quality*.
- Allows end users to integrate data *across* individual data sources without understanding SQL or schema mappings — but takes into account the query author’s perception of the authority or relevance of specific databases.

These requirements motivated us to design a new architecture for data integration and exchange, which we term the *collaborative data sharing system* or CDSS. The first implementation of the CDSS model has been realized in the ORCHESTRA system.

2 Collaborative Data Sharing Systems

The CDSS provides a principled semantics for exchanging data and updates among autonomous sites, which extends the data integration approach to encompass scientific data sharing practices and requirements — in a way that also generalizes to many other settings. The CDSS models the exchange of data among sites as *update propagation among peers*, which is subject to transformation (schema mapping), filtering (based on policies about source authority), and local revision or replacement of data.

Each participant or peer in a CDSS controls a local database instance, encompassing all data it wishes to manipulate. (In our ORCHESTRA implementation, this instance is stored in a conventional off-the-shelf DBMS.) Some of this data may have originated elsewhere. The participant normally operates in “disconnected” mode for a period, making local modifications to the data in the DBMS. As edits are made to this database, they are logged.

Publishing Updates to Data. At a peer administrator’s discretion, the *update exchange* capability of the CDSS is invoked, which first publishes the participant’s previously-invisible updates to “the world” at large. These updates published are permanently archived in the CDSS’s versioned storage system.

Importing Updates from Others. The second stage of update exchange then translates others’ updates to the peer’s local schema — also filtering which ones to apply, and reconciling any conflicts, according to the local administrator’s unique trust policies, before applying them to the local database. This update translation is based on an extension of data exchange [5]. Declarative *schema mappings* specify one participant’s schema-level relationships to other participants¹. Schema mappings may be annotated with *trust policies*: these specify filter conditions about *which* data should be imported to a given peer, as well as precedence levels for reconciling conflicts. Trust policies take into account the *provenance* or lineage [1,2,3,4,6] of data. If conflicts arise, they are *reconciled* based on an individual peer’s unique trust policies [12].

Querying across Peers. ORCHESTRA’s primary data sharing mechanisms are oriented around local data instances, and the user of any peer’s database may never need to

¹ These schema mappings may also include record linking tables translating terms or IDs from one database to another [10].

directly interact with ORCHESTRA. However, in some cases we would like to query *across* different peers, perhaps in different sub-fields. A scientist or other user in a CDSS may not know which peers are most relevant, nor how to write queries in SQL. ORCHESTRA's query system, **Q** [11], provides a facility through which non-expert users can author queries (or, more specifically, query templates that generate Web forms) over the relations on any peers in the system. **Q** is initially given a keyword query, which it attempts to match against schema elements. From the matching, it constructs a ranked list of potential queries that meet the user's information need, executes the top queries, and returns answers. The user may provide feedback on the answers, which are used to re-rank the queries and generate new, more relevant results. This last capability is a key factor not present in traditional keyword search over databases [8,9].

3 Conclusions

The collaborative data sharing paradigm represents a re-thinking of how data should be shared at large scale, when differences of opinion arise not only in the data representation, but also which data is correct. It defines new models and algorithms for transactional consistency, update exchange, provenance, and even ranking of keyword queries. The ORCHESTRA system represents an initial implementation of these ideas, which is being prototyped in several applications with biological collaborators, and which will soon be released to open source.

Acknowledgments

This work is funded by NSF IIS-0477972, 0513778, 0415810, and DARPA HR0011-06-1-0016. Development of the ORCHESTRA system and its foundations was in collaboration with (alphabetically) Todd Green, Sudipto Guha, Marie Jacob, Grigoris Karvounarakis, Fernando Pereira, Val Tannen, Partha Pratim Talukdar, and Nicholas Taylor. We thank Sarah Cohen-Boulakia, Olivier Biton, Sam Donnelly, and Renée Miller.

References

1. Benjelloun, O., Sarma, A.D., Halevy, A.Y., Widom, J.: ULDBs: Databases with uncertainty and lineage. In: VLDB (2006)
2. Buneman, P., Khanna, S., Tan, W.C.: Why and where: A characterization of data provenance. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, p. 316. Springer, Heidelberg (2000)
3. Chiticariu, L., Tan, W.-C.: Debugging schema mappings with routes. In: VLDB (2006)
4. Cui, Y.: Lineage Tracing in Data Warehouses. PhD thesis, Stanford University (2001)
5. Fagin, R., Kolaitis, P., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. *Theoretical Computer Science* 336, 89–124 (2005)
6. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: PODS (2007)

7. Halevy, A.Y., Ives, Z.G., Suciu, D., Tatarinov, I.: Schema mediation in peer data management systems. In: ICDE (March 2003)
8. Hristidis, V., Papakonstantinou, Y.: Discover: Keyword search in relational databases. In: VLDB (2002)
9. Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R., Karambelkar, H.: Bidirectional expansion for keyword search on graph databases. In: VLDB (2005)
10. Kementsietsidis, A., Arenas, M., Miller, R.J.: Mapping data in peer-to-peer systems: Semantics and algorithmic issues. In: SIGMOD (June 2003)
11. Talukdar, P.P., Jacob, M., Mehmood, M.S., Crammer, K., Ives, Z.G., Pereira, F., Guha, S.: Learning to create data-integrating queries. In: VLDB (2008)
12. Taylor, N.E., Ives, Z.G.: Reconciling while tolerating disagreement in collaborative data sharing. In: SIGMOD (2006)

Data Integration and Semantic Enrichment of Systems Biology Models and Simulations

Vijayalakshmi Chelliah, Lukas Endler, Nick Judy, Camille Laibe, Chen Li, Nicolas Rodriguez, and Nicolas Le Novère*

EMBL - European Bioinformatics Institute
Wellcome Trust Genome Campus
Hinxton, Cambridge CB10 1SD
United Kingdom
lenov@ebi.ac.uk

Abstract. The rise of Systems Biology approaches, in conjunction with the availability of numerous powerful and user-friendly modeling environments, brought computational models out of the dusty closets of theoreticians to the forefront of research in biology. Those models are becoming larger, more complex and more realistic. As any other type of data in life sciences, models have to be stored, exchanged and re-used. This was made possible by the development of a series of standards, that, when used in conjunction, can cover the whole life-cycle of a model, including the specification of its structure and syntax, the simulations to be run, and the description of its behaviour and resulting numerical output. We will review those standards, well-accepted or still under development, including the Minimal requirements (MIRIAM, MIASE), the description formats (SBML, SED-ML, SBRML) and the associated ontologies (SBO, KiSAO, TEDDY). We will show how their use by the community, through a rich toolkit of complementary software, can permit to leverage on everyone's efforts, to integrate models and simulations with other types of biological knowledge, and eventually lead to the fulfillment of one of Systems Biology's tenets of collaboration between biology, mathematics and computing science.

Keywords: database annotation, semantic annotation, data integration, quantitative models, computational systems biology, minimal requirement.

1 Introduction

Biological systems such as cellular, gene regulatory and protein interaction networks cannot be understood from studying the functions of their individual components. Systematic studies, considering all components at the same time, help precise understanding of the complex biological processes. Systems Biology focuses on the dynamics of biological processes at a "systems level", (i.e.) by considering interactions of all the components of the system. Computational Systems Biology deals with the construction of mathematical models that

* Corresponding author.

are central for handling the complex biological networks. Mathematical models describe systems in a quantitative manner on their own or in response to their environment and simulate their behaviour. Further, the progress in computational systems biology will lead to practical innovations in medicine and drug discovery.

Deriving a mathematical model, to simulate a biological process is a complex task. The modellers need to have information such as the kinetic law defining the rate of a reaction together with its parameters and experimental conditions, apart from the mathematical and biochemical knowledge. The quality and accuracy of quantitative models, that represent biological processes, depends mainly on the interaction between the experimentalists and the modellers. In recent years, application of modern computational and theoretical tools in modeling lead to an exponential increase both in number and complexity of quantitative models in biology. Modellers increasingly reuse and combine existing models. It often becomes impractical to re-implement models from literature. For easy and efficient use of the already published models, the models should be collected, curated and stored in a well structured databases, such as the BioModels Database [1], the Database of Quantitative Cellular Signalling (DOQCS) [2], JWS online [3] and the CellML Model repository [4] allowing users to search and retrieve models of interest. Recently, journals like Molecular Systems Biology, the BioMedCentral and PLoS journals encourage systems biologists to submit their coded models together with their papers. To enable researchers in quantitative systems biology to the existing models efficiently, a series of standards that includes the model format, the simulation to be run, the description of its behaviour and the resulting numerical outputs should be embedded in the model. In this review, we will discuss about the resources developed by the BioModels.net team in regard to minimum requirements, standard formats for encoding, and associated ontologies for 1) describing quantitative models, 2) describing simulation protocols and 3) describing simulation results (Figure 1).

The Minimal Information Requested In the Annotation of Models (MIRIAM; <http://www.ebi.ac.uk/miriam>) [5] defines the procedure for encoding and annotating models represented in machine-readable format. Minimum Information About a Simulation Experiment (MIASE; <http://www.ebi.ac.uk/compneur/srv/miase/>), describes the information needed to run and repeat a numerical simulation experiment derived from a given quantitative model. MIRIAM and MIASE are now, part of the Minimum Information for Biological and Biomedical Investigations project (MIBBI; <http://www.mibbi.org/>) [6]. MIBBI is a web-based, freely accessible resource for checklist projects, providing straight forward access to existing checklists (and to complementary data formats, controlled vocabularies, tools and databases), thereby enhancing both transparency and accessibility.

Biological ontologies play an important role in data integration. Three ontologies are developed by the BioModels.net project, in order to enrich the information provided with the models, to enhance integration of models, and integration of models with other types of knowledge. (1) The Systems Biology

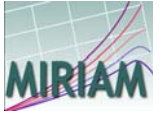
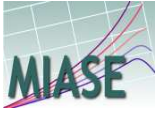




<i>Standard specification of quantitative models</i>	<i>Model description</i>	<i>Simulation description</i>	<i>Simulation results description</i>
<i>Minimal requirements</i>			?
<i>Data format</i>		SED-ML	SBRML
<i>Ontologies</i>			

Fig. 1. Standard specifications to encode quantitative systems biology models in three layers i.e 1) Model, 2) Simulation and 3) Simulation results description

Ontology (SBO; <http://www.ebi.ac.uk/sbo>) is a set of controlled vocabularies that add a semantic layer to the biochemical and mathematical description of a model, and act as a glue between different levels and types of representation. (2) The Kinetic Simulation Algorithm Ontology (KiSAO) classifies the approaches by model characteristics and numerical characteristics. Model characteristics include, for instance, the type of variables used for the simulation (such as discrete or continuous) and the spatial resolution. Numerical characteristics specify whether the systems' behaviour can be described as deterministic or stochastic, and whether the algorithms use fixed or adaptive time steps. (3) The Terminology for the Description of Dynamics (TEDDY), is a nascent effort to classify the behaviours of variables (eg. "oscillation", "bistable behaviour"), the characteristics of those behaviours (eg. "period", "bifurcation") and the functionalities of modules (eg. "negative feedback", "integrator"). The Open Biomedical Ontology (OBO) [7] consortium, works on expanding family of ontologies designed to be interoperable and logically well formed and to incorporate accurate representations of biological reality. SBO is available in OBO, OWL [8] and SBO-XML. KiSAO and TEDDY are available in OBO and OWL format respectively.

The most common format to encode quantitative models is SBML (Systems Biology Markup Language; <http://sbml.org/>) [9]. CellML [4] and NeuroML [10] are also used in the communities of physiology and computational neurobiology respectively. The Simulation Experiment Description Markup Language (SED-ML) [11] and the Systems Biology Results Markup Language (SBRML; <http://www.comp-sys-bio.org/static/SBRML-draft-21-08-2008.pdf>) are the formats developed for encoding the required simulation information and representing the simulation results, respectively.

2 Describing Quantitative Models

Quantitative models will be useful only if they can be accessed and reused easily by the scientific community. Most of the published quantitative models in biology are lost because they were not made available, not well formatted and characterised. To overcome this problem, the community had defined a set of guidelines for specifying quantitative models.

2.1 MIRIAM

To become part of a database (repository), quantitative models should be able to fulfil certain requirements and rules (MIRIAM: Minimal information requested in the annotation of models) [5]. These rules define procedures for encoding and annotating models represented in machine-readable form. The aim of MIRIAM is to define processes and schemes that will increase the confidence in model collections and enable the assembly of model collections of high quality. Firstly, the models must be 1) referred to a unique publication (that describes precisely the structure of the models, lists all quantitative parameters used, and describe the expected output), 2) encoded in a standard machine-readable format such as SBML or CellML, and 3) reproduce the results described in the reference publication. Secondly, the models must be annotated. The scheme of annotation includes assigning a name to the model, the details of the creators who encoded the model, date and time of creation and last modification of the model and links to external database resources. Model annotation and links to the external database resources are crucial features since they enhance model quality and are essential for search strategies. The data resources that are linked to could be, for instance, controlled vocabularies (Taxonomy, Gene Ontology, ChEBI etc.) or other databases (UniProt, KEGG, Reactome etc).

This annotation relates a piece of knowledge to a model constituent. The referenced information should be described using a triplet “data-type”, “identifier”, “qualifier”. The “data-type” is a unique, controlled, description of the type of data, written as a Uniform Resource Identifier [12]. The “identifier”, within the context of the “datatype”, points to a specific piece of knowledge. The “qualifier” is a string that serves to refine the relation between the referenced piece of knowledge and the described constituent. Example of qualifiers are “has a”, “is version of”, “is homologous to”, etc. Such a triplet can easily be exported later using RDF [13], to ease further automatic treatment.

2.2 SBO

Though there are many controlled vocabularies available that are used for quantitative models, several additional small controlled vocabularies are required to enable the systematic capture of information of the models. Thus Biomodels.net partners started developing their own ontology, Systems Biology Ontology (SBO).

The SBO provides additional semantics that can be used either to link formal representations of models to biological knowledge, or to interface different representations. SBO is currently composed of six different branches of vocabularies: quantitative parameter, participant type, modelling framework, mathematical expression, interaction and entity.

- quantitative parameter: This vocabulary includes terms such as “forward unimolecular rate constant”, “Hill coefficient”, “Michaelis constant”, that can be used to enhance SBML element parameter. In addition to the sub-classing links, a parameter can be defined in function of others through a mathematical construct.
- participant role: Includes participant functional type, like “catalyst”, “substrate”, “competitive inhibitor”, used for instance to enhance SBML element speciesReference, but also participant physical type, whether material, such as “macromolecule”, “simple chemical”, or conceptual, such as “gene” or “enzyme”, used to enhance SBML element species.
- modelling framework: Defines how to interpret a mathematical expression, such as “deterministic”, “stochastic”, “boolean” etc. This branch of SBO is only meant to state the context in which to interpret a mathematical expression, and is not meant to be redundant with KiSAO (for information about KiSAO, see below).
- mathematical expression: Classifies the construction used in biological modelling. In particular, it contains a taxonomy of kinetic rate equations. Examples of terms are “mass action kinetic”, “Henri-Michaelis-Menten kinetics”, “Hill equation” etc. The terms contain a mathematical construct that refers to the previous three vocabularies.
- interaction: Mutual or reciprocal action or influence that happens at a given place and time between participating entities and/or other interactions.
- entity: A real being, whether functional or material, that may participate in an interaction, a process or relationship of biological significance.

The branches are linked to the root by standard OBO has part relationships. Within a vocabulary, the terms are related by “is a” inheritances, which represent sub-classing, i.e. any instance of the child term is also an instance of the parent term. As a consequence, not only children are versions of the parents, but the mathematical expression associated with a child is a version of the mathematical expressions of the parents. In addition to its identifier and name, an SBO term contains a definition, a list of relationships, and optionally a mathematical construct, comments and synonyms.

Though SBO provide terms that are covered by certain existing ontologies, none of these ontologies provide features such as mathematical formulas corresponding to common biochemical rate laws expressed in ready-to-reuse MathML [\[14\]](#). Recent versions of the SBML specification (since Level 2 Version 2, [\[15\]](#)) allow model components to be annotated with SBO terms, therefore enhancing semantics of the model beyond the sole topology of interaction and mathematical expression. SBO is an open ontology, accessible in different format (OBO, OWL, SBO-XML) facilitating exchange and support by various tools, and accessible

programmatically via web services. SBO is a part of the OBO (Open BioMedical Ontologies). SBO, documentation and association resources are freely available at <http://www.ebi.ac.uk/sbo/>.

2.3 SBML

Systems Biology Markup Language (SBML), is a XML-based format for representing biochemical reaction networks. Though various formal languages were developed by different communities to encode models at different scales, the most successful format is SBML. The other common format to represent quantitative models is CellML, which is very similar to SBML. SBML is based on hierarchical lists of specified elements while CellML describes a model as a collection of linked generic components, offering the possibility of modular and multiscale models.

SBML can encode models consisting of biochemical entities linked by reactions to form biochemical networks. A model definition in SBML consists of lists of one or more of the following components:

- **Compartment:** A container of finite volume where the reacting entities may be located.
- **Species:** A chemical substance or entity that take part in the reaction.
- **Reaction:** A statement describing some transformation, transport or binding process that can change one or more species. Reactions have associated rate laws describing the manner in which they take place.
- **Parameter:** A quantity that has a symbolic name. SBML provides the ability to define parameters that are global to a model, as well as parameters that are local to a single reaction.
- **Unit definition:** A name for a unit used in the expression of quantities in a model. This is a facility for both setting default units and for allowing combinations of units to be given abbreviated names.
- **Rule:** A mathematical expression that is added to the model equations constructed from the set of reactions. Rules can be used to set parameter values, establish constraints between quantities, etc.

Most people like to look at the graphical representation of the reaction processes, since it gives the precise knowledge of the reaction processes. Graphical notations used by researchers and softwares are informal and are highly variable. So, the SBML community initiated a Systems Biology Graphical Notation (SBGN; <http://sbgn.org/>) project to help standardising a graphical notation for computational models in systems biology.

SBGN defines a comprehensive set of symbols with precise semantics, together with detailed syntactic rules defining their use and how diagrams are to be interpreted. The real payoff will come when researchers are as familiar with the notation as electronics engineers are with the notation of circuit schematics. SBML, SBO and MIRIAM together are useful for the generation of SBGN.

3 Describing Simulation Protocols

The models must be checked whether they can reproduce the results published in the reference scientific article. The simulation software packages read in a model expressed in SBML and translate it into their own internal format for model analysis. For instance, a package might provide the ability to simulate a model by constructing a set of differential equations representing the network and then performing numerical integration on the equations to explore the model's dynamic behaviour. The simulation software for simulating a model should be selected according to the features of the model, based on whether the model is, for example, a deterministic, continuous or stochastic discrete event model. Figure 2 illustrates the life cycle of quantitative models.

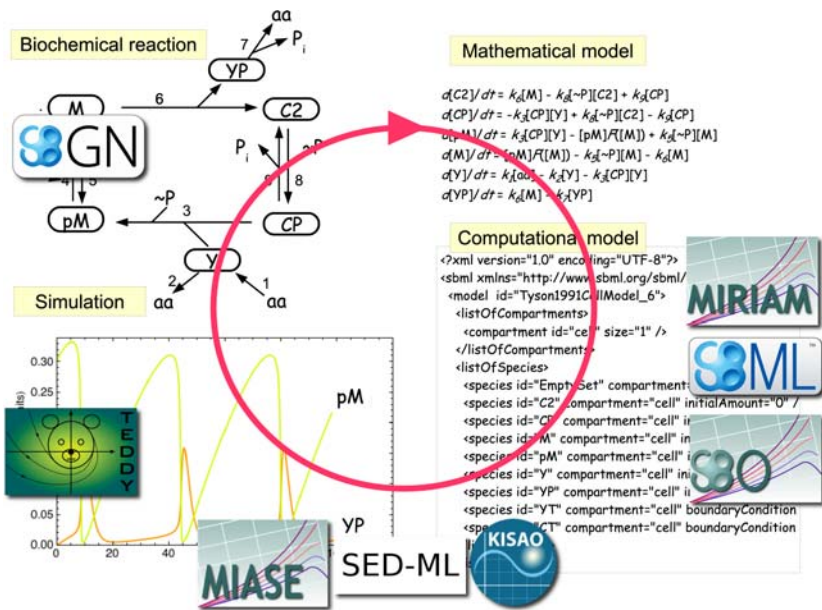


Fig. 2. Life cycle of systems biology models. The location of different logos represents their domain of relevance.

3.1 MIASE

The computational modeling procedure is not limited to the definition of the model structure. According to the MIRIAM specification, “the models must be able to reproduce the results published in the reference publication”. The guidelines to run the simulations would highly improve the efficient use of the models deposited in the repositories. This led to the development of the Minimum Information About a Simulation Experiment (MIASE; <http://www.ebi.ac.uk/>

compneur-srv/miase) project. It covers information about the simulated model, the simulation methods, the tasks performed and the output produced.

3.2 KiSAO

A crucial aspect of describing a simulation experiment is to precisely identify the simulation algorithm and simulation approach used for each step. So, it is important to have an ontology which can cover these details. The Kinetic Simulation Algorithm Ontology (KiSAO), characterise and categorise the existing simulation algorithm and approaches available to simulate the quantitative systems biology models. KiSAO is an open ontology, accessible in the OBO format, and is available from <http://www.ebi.ac.uk/comp-srv/miase/kiaso.html>.

3.3 SED-ML

The Simulation Experiment Description Markup Language (SED-ML) [11] is a formal XML based representation of models, aimed at encoding the simulation information required to implement MIASE guidelines. Each simulation tool that is capable of storing simulation settings uses its own internal storage format. For example, COPASI uses an XML based format for encoding the selected simulation algorithm and the task definitions. However, those formats are restricted to the specific simulation tool, and therefore simulation experiment descriptions cannot be exchanged with others. Similarly, the CellML community, in their CellML Metadata Specification [16] has planned to store simulation specification details inside the model definition and thus this approach is restricted to the CellML models only. SED-ML is independent of the software tools used to run the simulations. SED-ML can encode simulation experiments being run with several models, which can be in different formats (e.g. comparing simulation results of a CellML model and an SBML model). SED-ML can specify different simulation settings applicable to the same model (e.g. running a model with a stochastic and a deterministic simulation algorithm). Combination of both are also possible. It is easy to set up a simulation experiment that results in an output comparing a parameter of a CellML model to a parameter of an SBML model, depending on different simulation algorithms.

However, a number of important issues in simulation experiments is not currently covered. The description of more complex simulation tasks, for example, parameter scans, are not yet supported. Furthermore, currently SED-ML allows to combine variables from different tasks in one output - although the combinations depend on integrity restrictions. Another complex task, that is not yet supported is the linear execution of simulation experiments, meaning that the result of one simulation is used as the input for another simulation task. For example, the result of a steady state analysis will lead to a model with changed parameters. If that model then should be simulated using a time course simulation, the results of the steady state analysis have to be applied to the original model before. The definition of such sequences is not yet supported. Testing the implementation of SED-ML in different simulation tools will help enhancing the coverage and robustness of the format.

4 Describing Simulation Results

Given the computational model semantically-annotated with SBO, and a recipe for reproducing a simulation result with MIASE, there remains the problem of describing the observed behaviour in a systematic and machine-readable way.

4.1 TEDDY

At present, the observed simulation result is explained in a form of text accompanying the model (eg. “the time-course simulation demonstrates oscillations of variable x”). This kind of explanations, however are not clear and opaque to software tools. It would be useful, if one could compare the dynamic behaviour of systems. We must be able to answer questions such as: “How do I find a model containing protein X that displays periodic oscillations?”, “What behavioural features do all the models have in common?”, “Which model displays a behaviour matching my experimental data?” etc. To answer these questions an ontology that describes the dynamic behaviour of quantitative models is necessary. The Terminology for the Description of Dynamics (TEDDY) is an ontology which is designed to fulfil this. TEDDY classifies the behaviour of variables (eg. “oscillation”, “bistable behaviour”, the characteristics of those behaviour (eg. “period”, “bifurcation”) and the functionalities of modules (eg. “negative feedback”, “integrator”). TEDDY is encoded in OWL format (McGuinness and van Harmelen 2004) and available at <http://www.ebi.ac.uk/compneur-srv/teddy/>.

4.2 SBRML

Mostly quantitative systems biology models are encoded using SBML or CellML, which are community-wide accepted format and there are several hundreds of softwares that support SBML. However, there are no standard formats to represent simulation results. Dada et al., 2008 (<http://www.comp-sys-bio.org/static/SBRML-draft-21-08-2008.pdf>) have proposed a new markup language, Systems Biology Results Markup Language (SBRML), which is complementary to SBML. SBRML is XML based markup language, which is intended to specify results from operations carried out on models. The initial state of a biochemical reaction network is defined in the SBML model. To simulate and analyse the reaction network, a software package takes the SBML model as input and transforms the initial state through an operation that the software implements. The outcome of an operation is a new state of the system. The new states, which may be single or multiple states, are captured and described in SBRML. The result of an operation consists of one or more result components. Software tools implement an operation using a specific algorithm. The type of algorithm used in an operation and the type of operation (e.g. steady state, time course, parameter scan, parameter estimation, etc.) are captured in an ontology term definition. SBRML is intended to allow any type of systems biology results to be represented. It is currently structured as follows: On the top level, SBRML consists of ontology terms, mode, and operations. The second level describes each operation application, which consists of software, algorithm

and result. The next level is the content of the result, which consists of one or more result components. Each result component consists of the description of the data being represented and data itself.

5 Conclusions

A well-curated and annotated model will support reuse, reliability and validation for a range of users, both human and machine. Appropriate curation with semantic enrichment improves the quality of the models, facilitating data integration that is crucial for the efficient usage and analysis of the models. Integrated data enables the user to effectively access a consistent range of data from a single location. As the methodology for studying system biology matures, the rules for creating models will have to become stricter in order to ensure the quality of the models as well as the value of the repository of quantitative models. Also, more and more software tools hopefully will support common open formats and make use of standardized annotations and allow their manipulation.

Acknowledgements

BioModels.net activities are supported by the European Molecular Biology Laboratory and the National Institute of General Medical Sciences. MIRIAM resources benefited from support by the Biotechnology and Biological Sciences Research Council, SBO by the EU ELIXIR preparatory phase and SBGN by the New Energy and Industrial Technology Development Organization.

References

1. Le Novère, N., Bornstein, B., Broicher, A., Courtot, M., Donizelli, M., Dharuri, H., Li, L., Sauro, H., Schilstra, M., Shapiro, B., Snoep, J.L., Hucka, M.: BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Res.* 34, D689–D691 (2006)
2. Sivakumaran, S., Hariharaputran, S., Mishra, J., Bhalla, U.S.: The Database of Quantitative Cellular Signaling: management and analysis of chemical kinetic models of signaling networks. *Bioinformatics* 19, 408–415 (2003)
3. Olivier, B.G., Snoep, J.L.: Web-based kinetic modelling using JWS Online. *Bioinformatics* 20, 2143–2144 (2004)
4. Lloyd, C.M., Halstead, M.D.B., Nielsen, P.F.: CellML: its future, present and past. *Prog. Biophys. Mol. Biol.* 85, 433–450 (2004)
5. Le Novère, N., Finney, A., Hucka, M., Bhalla, U.S., Campagne, F., Collado-Vides, J., Crampin, E.J., Halstead, M., Klipp, E., Mendes, P., Nielsen, P., Sauro, H., Shapiro, B., Snoep, J.L., Spence, H.D., Wanner, B.L.: Minimum information requested in the annotation of biochemical models (MIRIAM). *Nat. Biotechnol.* 23, 1509–1515 (2005)

6. Taylor, C.F., Field, D., Sansone, S.A., Aerts, J., Apweiler, R., Ashburner, M., Ball, C.A., Binz, P.A., Bogue, M., Booth, T., Brazma, A., Brinkman, R.R., Clark, A.M., Deutsch, E.W., Fiehn, O., Fostel, J., Ghazal, P., Gibson, F., Gray, T., Grimes, G., Hancock, J.M., Hardy, N.W., Hermjakob, H., Julian, R.K., Kane, M., Kettner, C., Kinsinger, C., Kolker, E., Kuiper, M., Le Novère, N., Leebens-Mack, J., Lewis, S.E., Lord, P., Mallon, A.M., Marthandan, N., Masuya, H., McNally, R., Mehrle, A., Morrison, N., Orchard, S., Quackenbush, J., Reecy, J.M., Robertson, D.G., Rocca-Serra, P., Rodriguez, H., Rosenfelder, H., Santoyo-Lopez, J., Scheuermann, R.H., Schober, D., Smith, B., Snape, J., Stoeckert, C.J., Tipton, K., Sterk, P., Untergasser, A., Vandesompele, J., Wiemann, S.: Promoting coherent minimum reporting guidelines for biological and biomedical investigations: the MIBBI project. *Nat. Biotechnol.* 26, 889–896 (2008)
7. Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L.J., Eilbeck, K., Ireland, A., Mungall, C.J., Consortium, O.B.I., Leontis, N., Rocca-Serra, P., Ruttenberg, A., Sansone, S.A., Scheuermann, R.H., Shah, N., Whetzel, P.L., Lewis, S.: The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nat. Biotechnol.* 25, 1251–1255 (2007)
8. Smith, M., Welty, C., McGuinness, D.: OWL Web Ontology Language Reference. Technical report, W3C (2004)
9. Hucka, M., Finney, A., Sauro, H.M., Bolouri, H., Doyle, J.C., Kitano, H., Arkin, A.P., Bornstein, B.J., Bray, D., Cornish-Bowden, A., Cuellar, A.A., Dronov, S., Gilles, E.D., Ginkel, M., Gor, V., Goryanin, I.I., Hedley, W.J., Hodgman, T.C., Hofmeyr, J.H., Hunter, P.J., Juty, N.S., Kasberger, J.L., Kremling, A., Kummer, U., Le Novère, N., Loew, L.M., Lucio, D., Mendes, P., Minch, E., Mjolsness, E.D., Nakayama, Y., Nelson, M.R., Nielsen, P.F., Sakurada, T., Schaff, J.C., Shapiro, B.E., Shimizu, T.S., Spence, H.D., Stelling, J., Takahashi, K., Tomita, M., Wagner, J., Wang, J., Forum, S.B.M.L.: The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* 19, 524–531 (2003)
10. Goddard, N., Hucka, M., Howell, F., Cornelis, H., Skankar, K., Beeman, D.: Towards NeuroML: Model Description Methods for Collaborative Modeling in Neuroscience. *Phil. Trans. Royal Society series B* 356, 1209–1228 (2001)
11. Köhn, D., Le Novère, N.: SED-ML – An XML Format for the Implementation of the MIASE Guidelines. In: Heiner, M., Uhrmacher, A.M. (eds.) CMSB 2008. LNCS (LNBI), vol. 5307, pp. 176–190. Springer, Heidelberg (2008)
12. Berners-Lee, T., Fielding, R., Masinter, L.: Uniform Resource Identifier (URI): Generic Syntax, <http://www.gbiv.com/protocols/uri/rfc/rfc3986.html>
13. Beckett, D.: RDF/XML Syntax Specification (Revised). Technical report, W3C (2004)
14. Ausbrooks, R., Buswell, S., Carlisle, D., Dalmas, S., Devitt, S., Diaz, A., Froumentin, M., Hunter, R., Ion, P., Kohlhase, M., Miner, R., Poppelier, N., Smith, B., Soiffer, N., Sutor, R., Watt, S.: Mathematical Markup Language (MathML) Version 2.0, 2 edn. Technical report, W3C (2003)
15. Hucka, M., Finney, A., Le Novère, N.: Systems Biology Markup Language (SBML) Level 2: Structures and Facilities for Model Definitions. Technical report (2006)
16. Miller, A.: CellML Simulation metadata specification - a specification for simulation metadata. Technical report, Auckland Bioengineering Institute (2007)

Linking Life Sciences Data Using Graph-Based Mapping

Jan Taubert¹, Matthew Hindle¹, Artem Lysenko¹, Jochen Weile², Jacob Köhler³,
and Christopher J. Rawlings¹

¹ Department of Biomathematics and Bioinformatics, Rothamsted Research, Harpenden, UK

² School of Computing Science, Newcastle University, Newcastle upon Tyne, UK

³ Protein Research Group, University of Tromsø, Norway
{jan.taubert, matthew.hindle, artem.lysenko,
chris.rawlings}@bbsrc.ac.uk,

j.weile@ncl.ac.uk, Jacob.Kohler@fagmed.uit.no

Abstract. There are over 1100 different databases available containing primary and derived data of interest to research biologists. It is inevitable that many of these databases contain overlapping, related or conflicting information. Data integration methods are being developed to address these issues by providing a consolidated view over multiple databases. However, a key challenge for data integration is the identification of links between closely related entries in different life sciences databases when there is no direct information that provides a reliable cross-reference. Here we describe and evaluate three data integration methods to address this challenge in the context of a graph-based data integration framework (the ONDEX system). A key result presented in this paper is a quantitative evaluation of their performance in two different situations: the integration and analysis of different metabolic pathways resources and the mapping of equivalent elements between the Gene Ontology and a nomenclature describing enzyme function.

1 Introduction

A major challenge for users of data from genetics, genomics and other high throughput experimental methods used for systems biological research is how to interpret their results in the wider context of the biological information that is held in the many databases scattered around the internet. Furthermore, some of these databases have similar or overlapping coverage and the user is faced with the additional challenge of generating a consensus data set or selecting the “best” data source. The effective integration of information from different biological databases and information resources is recognized as a pre-requisite for many aspects of systems biological research and has been shown to be advantageous in a wide range of use cases such as the analysis and interpretation of ‘omics data [1], biomarker discovery [2] and the analysis of metabolic pathways for drug discovery [3].

Software designed to integrate data for life sciences applications has to address two classes of problem. It must provide a general solution to the **technical heterogeneity**, which arises from the different data formats, access methods and protocols used by different databases. More significantly, it must address the **semantic heterogeneities**

evident in a number of life science databases. The most challenging source of semantic heterogeneity comes from the diversity and inconsistency among naming conventions for genes, gene functions, biological processes and structures among different species (or even within species). In recent years, significant progress in documenting the semantic equivalence of terms used in the naming of biological concepts and parts has been made in the development of a range of biological ontology databases which are coordinated under the umbrella of organisations such as the Open Biomedical Ontologies Foundry (www.obofoundry.org).

Data Integration

Software tools that solve aspects of the data integration problem have been developing for some time. The early approaches, which produced popular software such as SRS [4] used indexing methods to link documents or database entries from different databases and providing a range of text and sequence-based search and retrieval methods for users to assemble related data sets. The methods used by SRS (and related tools) address what we describe as the technical integration challenge.

More recently, data integration approaches have been developed that “drill down” in to the data and seek to link objects at a more detailed level of description. Many of these approaches exploit the intuitively attractive representation of data as graphs or networks with nodes representing things or concepts and edges representing how they are related or connected. For example a metabolic pathway could be represented by a set of nodes identifying the metabolites linked by edges representing enzyme reactions.

Data integration systems that exploit graph based methods include PathSys [5] or BN++ [6] and the ONDEX system [1, 7]. Both BN++ and ONDEX are available as open source software. The methods presented in this publication are implemented as part of the ONDEX system and their quantitative evaluation helps maximise the integrity of the ONDEX integration process.

The integration of data generally follows three conceptual stages as illustrated in Figure 1: (1) normalising into a generic data structure in order to overcome predominantly technical heterogeneities between data exchange formats; (2) identifying equivalent and related entities among the imported data sources to overcome semantic heterogeneities at the entry level, and (3) the data analysis, information filtering and knowledge extraction.

In order to make the ONDEX system as extensible as possible, we have taken particular steps to separate the second stage both conceptually and practically. Our motivations for doing this are that we wish to: preserve original relationships and metadata from the source data; make this semantic integration step easily extensible with new methods; implement multiple methods for recognising equivalent data concepts to enhance the quality of data integration; support reasoning methods that make use of the information generated in this step to improve the quality of data integration and facilitate the reuse of our methods. ONDEX therefore provides a convenient platform for evaluating different data integration methods which could be subsequently exploited by other software platforms as part their data integration workflows (e.g. Cytoscape [8], SemanticWebFolders [9]).

Our hypothesis is that multiple methods for semantic data integration are necessary because of ambiguities and inconsistencies in the source data that will require

different treatment depending on the source databases. In many cases, exact linking between concepts through unique names will not always be possible and therefore mappings will need to be made using inexact methods. Unless these inexact methods can be used reliably, the quality of the integrated data will be degraded.

In this paper we present a quantitative evaluation of three data integration algorithms that are used in the second data integration step (Figure 1): *Accession based mapping*, *Synonym mapping* and an algorithm called *StructAlign*. These all use concept names in different ways to assign equivalences. The approach used in the first two methods will be relatively obvious. The *StructAlign* algorithm, however, combines both information retrieval approaches and graph based analysis and will therefore be described in more detail. The three algorithms were chosen as progressively more flexible concept matching methods that were considered suitable for life science databases where concept names play a key role in defining semantics. A novel aspect of this paper is that the algorithms are evaluated in two representative contexts: the integration and analysis of metabolic pathways, and the mapping of equivalent elements in ontologies and nomenclatures.

To calibrate our data integration methods with well structured data, we use the mapping of equivalent elements from the ontologies and nomenclatures extracted from the ENZYME [11] and GO [12] databases. To evaluate mapping methods in a more challenging integration task we use the creation of an integrated database from two important biological pathway resources, the KEGG [13] and AraCyc [14] databases.

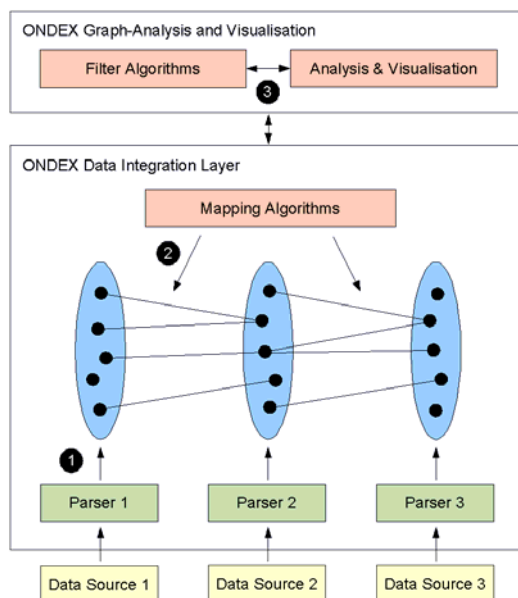


Fig. 1. Data integration in ONDEX consists of 3 steps. 1) Importing and converting data sources into the generic data structure of ONDEX, 2) Linking of equivalent or related entities within the different data sources, 3) Knowledge extraction in the graph based analysis component.

2 Methods

2.1 Data Import and Export

Following Figure 1, the first data integration step loads and indexes data from different sources. ONDEX provides several options for loading data into the internal data warehouse and a range of parsers have been written for commonly used data sources and exchange formats, see <http://ondex.sourceforge.net/feats.php>. In addition users can convert their data into an ONDEX specific XML or RDF dialect for which generic parsers [15] are provided.

The role of all parsers is to load data from different data sources into the generic data structure used in the ONDEX data warehouse. In simple terms, this generic data structure can be seen as a graph, in which concepts are the nodes and relations are the edges. By analogy with the use of ontologies for knowledge representation in computer science, concepts are used to represent real-world objects [16]. Relations are used to represent the different ways in which concepts are connected to each other. Furthermore, concepts and relations may have additional properties and optional characteristics attached to them. A formal description of the ONDEX graph data structure is presented in [1] and an XML and RDF representation is described in [15].

During the import process, names for concepts are lexicographically normalised by replacing non-alphanumeric characters with white spaces so that only numbers and letters are kept in the name. In addition, consistency checks are performed to identify, for example, empty or malformed concept names.

Datasets integrated in the ONDEX warehouse can be exported into several standard exchange formats such as FASTA, GML, GraphML, RDF, SBML, XGMML and XML. They can also be accessed programmatically through the API of the ONDEX system.

2.2 Data Integration Methods and Algorithms

The second data integration step (following Figure 1) links equivalent and related concepts and therefore creates relations between concepts from different data sources. Different combinations of mapping methods can be used to create links between equivalent or related concepts. Rather than merging elements that are found to be equivalent, the mapping methods create a new equivalence relation between such concepts. Each mapping method can be made to create a score value reflecting the belief in a particular mapping and information about the parameters used. These scores are assigned as edge weights to the graph and form the foundation for the statistical analysis presented later. The information on edges additionally enables the user to track evidence for why two concepts were mapped by a particular mapping method.

We now describe the different mapping methods that will be evaluated in the later sections of this paper. This paper provides an extended and more formal account of these mapping methods, which were first mentioned without elaboration in [7] and [10] and this is the first time that a detailed evaluation has been reported.

Accession based mapping: Most of the well-structured and publicly managed repositories of life science data use an accession coding systems to uniquely identify individual database entries. These codes are persistent over database versions. Cross references between databases of obviously related data (e.g. protein and DNA sequences) can generally be found using accession codes and these can be easily exploited to link related concepts. Accession codes may not always present a one-to-one relationship between entries of different databases. References presenting one-to-many relationships are called ambiguous. Accession codes are indexed for better performance during information retrieval. Accession based mapping by default uses non-ambiguous accessions to create links between equivalent concepts, i.e. concepts that share the same references to other databases in a one-to-one relationship. It is, however, possible to take one-to-many relationships into account for accession based mapping but this feature was not evaluated in this study because accession based mapping was required to establish a “gold standard” between data sources to identify true pairs of equivalent entries.

Synonym mapping: Entries in biological databases often have one or more human-readable names, e.g. gene-names. Depending on the data source, some of these names will be exact synonyms, such as the chemical name of an enzyme; others only related synonyms, such as a term for enzymatic function. Exact synonyms are flagged as preferred concept names during the import process. Related synonyms are added to concepts as additional concept names and concept names are indexed for better performance during information retrieval. The default method for synonym mapping creates a link between two concepts if two or more preferred concept names are matching. As a simple example of such ambiguity, the term “mouse” shows that consideration of only one synonym is not usually enough for the disambiguation of the word, i.e. “mouse” can mean computer mouse or the rodent *Mus musculus*. The threshold for the number of synonyms to be considered a match is a parameter for the synonym mapping method.

StructAlign mapping: In some cases, two or more synonyms for a concept are not available in the integrated data. To disambiguate the meaning of a synonym shared by two concepts, the *StructAlign* mapping algorithm considers the graph neighbourhood of such concepts. A breadth-first search of a given depth (≥ 1) starting at each of the two concepts under consideration yields the respective reachability list for each concept. *StructAlign* processes these reachability lists and searches for synonym matches of concepts at each depth of the graph neighbourhood. If at any depth one or more pairs of concepts which share synonyms are found, *StructAlign* creates a link between the two concepts under consideration.

Pseudo-code for StructAlign (see Figure 2): Let O denote the generic graph structure consisting of a set of concepts $C(O)$ and a set of relations $R(O) \subseteq C(O) \times C(O)$. Every concept $c \in C(O)$ has two additional attributes assigned: a) a concept class $cc(c)$ characterising the type of real world entity represented by the concept (e.g. a gene), b) a controlled vocabulary $cv(c)$ stating the data source (e.g. KEGG) the concept was extracted from. Every relation $r \in R(O)$ is a tuple $r = (f, t)$ with f the “from”-concept and t the “to”-concept of the relation.

```

index ← indexed list of concept names
cutoff ← maximal depth of graph neighbourhood
function StructAlign(O, index, cutoff) {
  matchesMap ← new map of concepts to concept lists
  // search for concept name matches
  for all  $c \in C(O)$  do
    for all name  $\in cn(c) | preferred(name) = true$  do
      matches ← index.search(name)
      for all  $c' \in matches$  with  $cc(c) = cc(c') \wedge cv(c) \neq cv(c')$  do
        matchesMap.add( $c, c'$ )
  connectedMap ← new map of concepts to concept lists
  // calculate direct neighbourhood
  for all  $r \in R(O)$  with  $r = (f, t)$  do
    if  $cv(f) = cv(t) \wedge f \neq t$  do
      connectedMap.add( $f, t$ )
      connectedMap.add( $t, f$ )
  reachabilityMap ← clone(connectedMap)
  // modified breadth first search with depth cutoff
  for all  $i \in [1..cutoff]$  do
    for all  $(x, (y_1 \dots y_n)) \in reachabilityMap$  do
      for all  $j \in [1..n]$  do
        reachabilityMap.add( $x, connectedMap.get(y_j)$ )
  // look at neighbourhood of bidirectional matches
  for all  $(a, (b_1 \dots b_n)), (b_i, (c_1 \dots c_m)) \in matchesMap | a \in (c_1 \dots c_m)$  do
    na ← reachabilityMap.get( $a$ )
    nb ← reachabilityMap.get( $b_i$ )
    for all  $x \in na$  do
      if  $\exists y \in matchesMap.get(x) | y \in nb$  do
        O.createRelation( $a, b_i$ )
}

```

Fig. 2. Pseudo-code for StructAlign algorithm

Assuming the search for a concept name in the indexed list of concept names takes constant time, the algorithm has a worst case runtime of $O(c^3 * \log(c))$ on dense graphs. Its average runtime on sparse graphs is $O(c^2)$.

2.3 Graph Analysis and Visualisation

Data integration involving large data sets can create very large networks that are densely connected. To reduce the complexity of such networks for the user, information filtering, network analysis and visualisation (see Figure 1, step 3) is provided in a client application for ONDEX called OVTK [1]. The combination of data integration

and graph analysis with visualisation has been shown to be valuable for a range of data integration projects in different domains, including microarray data analysis [1], support of scientific database curation [17, 18] and scoring the quality of terms and definitions in ontologies such as the Gene Ontology [19].

A particularly useful feature in the OVTK is to visualise an overview of the types of data that have been integrated in the ONDEX system. This overview is called the ONDEX meta-graph. It is generated as a network based on the generic data structure used in ONDEX, which contains a type system for concepts and relations. Concepts are characterised using a class hierarchy and relations have an associated type. This information about concept classes and relation types is visualised as a graph with which the user can interact to specify semantic constraints – such as changing the visibility of concepts and relations in the visualisation and analysis of the data network structure.

2.4 Evaluation Methods

The mapping algorithms presented in Section 2.2 can be configured using different parameters. According to the selection of the parameters these methods yield different mapping results. To evaluate their behaviour, we used two different test scenarios: the mapping of equivalent elements in ontologies and the integration and analysis of metabolic pathways.

The evaluation of a mapping method requires the identification of a reference data set, sometimes also referred to as “gold standard”, describing the links that really exist between data and that can be compared with those which are computed. Unfortunately, it is rare that any objective definition of a “gold standard” can be found when working on biological data sets and so inevitably most such evaluations required the development of expertly curated data sets. Since these are time-consuming to produce, they generally only cover relatively small data subsets and therefore the evaluation of precision and recall is inevitably somewhat limited.

In Section 3.1 we describe the results of mapping together two ontologies; namely the Enzyme Commission (EC) Nomenclature [11] and Gene Ontology (GO) [12]. In this case, the Gene Ontology project provides a manually curated mapping to the Enzyme Nomenclature called **ec2go**. We have therefore selected **ec2go** as the first of our gold standards. We also consider the cross references between the two ontologies contained in the integrated data as the second gold standard for this scenario.

Section 3.2 presents the results from the evaluation of a mapping created between the two metabolic pathway databases KEGG and AraCyc. Unfortunately, a manually curated reference set is not available for this scenario. We must therefore rely on the cross references between the two databases that we can calculate through our accession based mapping as the nearest equivalent of a gold standard for this scenario.

3 Results

We have evaluated our mapping algorithms using the standard measures of precision (Pr), recall (Re) and F_1 -score [20]:

$$\text{Pr} = \frac{tp}{tp + fp} \quad \text{Re} = \frac{tp}{tp + fn} \quad F_1 = \frac{2 \cdot \text{Pr} \cdot \text{Re}}{\text{Pr} + \text{Re}}$$

The accession based mapping algorithm (Acc) was used with default parameters, i.e. only using non-ambiguous accessions. This choice has been made to obtain a “gold-standard” through accession based mapping, i.e. increasing the confidence in the relations created. When evaluating the synonym mapping (Syn) and StructAlign (Struct) algorithms, parameters were varied to examine the effect of the number of synonyms that must match for a mapping to occur. This is indicated by the number in brackets after the algorithm abbreviation (e.g. Struct(1)). We also evaluated a second variant of each algorithm in which related synonyms of concepts were used to find a mapping. The use of this algorithmic variant is indicated by an asterisk suffix on the algorithm abbreviation (e.g. Syn(1)*).

3.1 Mapping Methods – Enzyme Nomenclature vs. Gene Ontology

In this evaluation our goal was to maximise the projection of the Enzyme Commission (EC) nomenclature onto the Gene Ontology. This would assign every EC term one or more GO terms. For the evaluation we used **ec2go** (revision 1.67) and `gene_ontology_edit.obo` (revision 5.661) obtained from `ftp://ftp.geneontology.org`. We also downloaded `enzclass.txt` (last update 2007/06/19) and `enzyme.dat` (release of 2008/01/15) from `ftp://ftp.expasy.org`. The data files were parsed into the ONDEX data network and the mapping algorithms applied using the ONDEX integration pipeline. To determine the optimal parameters for this particular application case we systematically tested different combinations of the mapping algorithms with the variants and parameter options as described above. Table 1 summarises the mapping results and compares the performances with the “gold standard” data sets from **ec2go** and by accession mapping (Acc).

Table 1. Mapping results for Enzyme Nomenclature to Gene Ontology

Method	TP,FP ec2go	TP,FP Acc	Pr,Re [%] ec2go	Pr,Re [%] Acc	F ₁ -score ec2go	F ₁ -score Acc
Ec2go	8063 , 0	8049 , 14	100 , 100	99 , 84	100	91
Acc	8049 , 1441	9490 , 0	84 , 99	100 , 100	91	100
Syn(1)	7460 , 934	7462 , 932	88 , 92	88 , 77	90	82
Syn(1)*	7605 , 2581	7606 , 2580	74 , 94	74 , 80	83	77
Syn(2)*	4734 , 374	4738 , 370	92 , 58	92 , 49	71	64
Syn(3)*	2815 , 117	2816 , 116	96 , 34	96 , 29	51	45
Struct(1)	1707 , 63	1712 , 58	96 , 21	96 , 18	34	30
Struct(1)*	1761 , 279	1766 , 274	86 , 21	86 , 18	34	30
Struct(2)	7460 , 934	7462 , 932	88 , 92	88 , 77	90	82
Struct(2)*	7605 , 2581	7606 , 2580	74 , 94	74 , 80	83	77
Struct(3)	7460 , 934	7462 , 932	88 , 92	88 , 77	90	82
Struct(3)*	7605 , 2581	7606 , 2580	74 , 94	74 , 80	83	77

ec2go = imported mapping list (1st gold standard), Acc = Accession based mapping (2nd gold standard), Syn = Synonym mapping, Struct = StructAlign, * = allow related synonyms, TP = True Positives, FP = False positives, Pr = Precision, Re = Recall, F₁-score. Synonym mapping was parameterised with a number that states how many of the names had to match to create a link between concepts. StructAlign was parameterised with the depth of the graph neighbourhood.

The first two rows of Table 1 show the performance of our “gold standard” methods tested against themselves. As can be seen by reviewing the F_1 scores in the subsequent rows of Table 1, the most accurate synonym mapping requires the use of just one synonym. It does not help to search for further related synonyms (Syn(1,2,3)*). The explanation for this is that the EC nomenclature does not distinguish between exact and related synonyms. Therefore, concepts belonging to the EC nomenclature have only one preferred concept name (exact synonym) arbitrarily chosen to be the first synonym listed in the original data sources. A large number of entries in the EC nomenclature only have one synonym described, which probably explains the low recall of Syn(2)* and Syn(3)*.

The use of the more complex StructAlign algorithm, which uses the local graph topology to identify related concepts has low recall when only a single synonym is required to match (Struct(1) and Struct(1)*). This almost certainly results from differences in graph topology between EC nomenclature and Gene Ontology. The Gene Ontology has a more granular hierarchy, i.e. there is more than one hierarchy level between two GO terms mapped to EC terms, whereas the EC terms are only one hierarchy level apart. As the StructAlign search parameters are increased, more of the graph context is explored and accordingly we see improved F_1 scores.

The highest F_1 -scores come from Syn(1), Struct(2) and Struct(3) respectively. Allowing the related synonyms into the search (the * algorithm variants) did not improve precision. Neither did extending the graph neighbourhood search depth from Struct(2) to Struct(3) as all the neighbourhood matches had already been found within search depth 2.

During the integration of these datasets for this evaluation we observed some inconsistencies in the **ec2go** mapping list. The identification of such data quality issues are often a useful side-effect of developing integrated datasets. The inconsistencies we identified are listed in Table 2 and were revealed during the import of the **ec2go** data file after preloading the Gene Ontology and EC nomenclature into ONDEX.

Table 2. Inconsistencies in ec2go

Accession	Mapping	Reason for failure
GO:0016654	1.6.4.-	Enzyme class does not exist, transferred entries
GO:0019110	1.18.99.-	Enzyme class does not exist, transferred entries
GO:0018514	1.3.1.61	Enzyme class does not exist, deleted entry
2.7.4.21	GO:0050517	GO term obsolete
GO:0047210	2.4.1.112	Enzyme class does not exist, deleted entry
1.1.1.146	GO:0033237	GO term obsolete
GO:0016777	2.7.5.-	Enzyme class does not exist, transferred entries
GO:0004712	2.7.112.1	Enzyme class does not exist, possible typo
2.7.1.151	GO:0050516	GO term obsolete

Every inconsistency was checked by hand against gene_ontology_edit.obo, enzclass.txt and enzyme.dat.

We assume that most of the problems are due to the parallel development of both ontologies, i.e. GO references that were transferred or EC entries being deleted or vice versa. A few of the inconsistencies were possible typing errors. It remains a possibility that other “silent” inconsistencies remain in **ec2go** that these integration methods would not find.

An example of a plausible “silent” inconsistency that we could not detect might result from changing the last figure of a term in the EC nomenclature from 2.4.1.112 to 2.4.1.11. This remains a well-formed EC nomenclature term and could not be detected without other information being available to identify the error.

3.2 Mapping Methods – KEGG vs. AraCyc

The KEGG and AraCyc pathway resources are both valuable for biologists interested in metabolic pathway analysis. Due to the different philosophies behind these two databases [21], however, they do have differences in their contents. Plant scientists wishing to work with biochemical pathway information would therefore benefit from an integrated view of AraCyc and the *A. thaliana* subset of KEGG and so this makes a realistic test. These two databases were chosen for this evaluation because both pathway databases annotate metabolites and enzymes in the pathways with standardised CAS registry numbers [22] and ATG numbers [23] respectively. It is therefore possible to evaluate the precision, recall and F₁-score of the different mapping methods using accession based mapping between these registry numbers as a “gold standard”.

For our evaluation we used the KEGG database (release 46) obtained from <ftp://ftp.genome.jp> and the AraCyc database (release 4.1) obtained from <http://www.arabidopsis.org/biocyc/index.jsp>. The AraCyc database contained 1719 metabolites and 6192 enzymes. The KEGG database contained 1379 metabolites and 1126 enzymes. The evaluation results from the mapping between metabolites from these two databases is summarised in Table 3.

Table 3. Mapping results for KEGG and AraCyc databases – Metabolites

Method	TP	FP	Pr [%]	Re [%]	F ₁ -score
Acc	662	0	100	100	100
Syn(1)	516	726	41	77	54
Syn(1)*	600	1038	36	90	52
Syn(2)	14	0	100	2	4
Syn(2)*	396	348	53	59	56
Syn(3)*	190	114	62	28	39
Struct(2)	374	448	45	56	50
Struct(2)*	416	614	40	62	49
Struct(3)	382	470	44	57	50
Struct(3)*	432	670	39	65	48

Acc = Accession based mapping (gold standard), Syn = Synonym mapping, Struct = StructAlign, * = allow related synonyms, TP = True Positives, FP = False positives, Pr = Precision, Re = Recall, F₁-score. Synonym mapping was parameterised with a number that states how many of the names had to match to create a link between concepts. StructAlign was parameterised with the depth of the graph neighbourhood.

Accession based mapping between metabolite names found 662 out of 1379 possible mappings. A closer look reveals that CAS registry numbers are not always assigned to metabolite entries. Therefore, the accession based mapping misses possible links and cannot be used naively as a gold standard for this particular application case. In this evaluation, accession based mapping underestimates possible mappings, which leads to low precision for synonym mapping and StructAlign. We manually reviewed a random set of the false positive mappings returned by Syn(1) and Struct(3) and this

revealed that most of the mappings were correct and metabolites shared similar chemical names. Subject to further investigation, this example shows that relying only on accession based data for integration might miss out some important links between data sources.

The evaluation result from the mapping between enzyme names from KEGG and AraCyc is summarised in Table 4.

Table 4. Mapping results for KEGG and AraCyc databases – Enzymes

Method	TP	FP	Pr [%]	Re [%]	F ₁ -score
Acc	2240	0	100	100	100
Syn(1)	282	0	100	12	22
Syn(1)*	2234	344	86	99	92
Syn(2)*	144	2	98	6	12
Syn(3)*	6	0	100	1	1
Struct(1)	234	0	100	10	18
Struct(1)*	2134	0	100	95	97
Struct(2)	282	0	100	12	22
Struct(2)*	2232	336	86	99	92
Struct(3)	282	0	100	12	22
Struct(3)*	2234	336	86	99	92

Acc = Accession based mapping (gold standard), Syn = Synonym mapping, Struct = StructAlign, * = allow related synonyms, TP = True Positives, FP = False positives, Pr = Precision, Re = Recall, F₁-score. Synonym mapping was parameterised with a number that states how many of the names had to match to create a link between concepts. StructAlign was parameterised with the depth of the graph neighbourhood.

The accession based mapping between enzymes uses the ATG numbers available in both KEGG and AraCyc. Entries from KEGG can be labelled with two or more ATG numbers representing multiple proteins involved in the enzymatic function, whereas AraCyc entries usually only have one ATG number. This results in one-to-many hits between KEGG and AraCyc explaining why a total of 2240 instead of only 1126 mapping were found. This is a good example of how the differences in the semantics between biological databases make it difficult to define a gold standard for evaluating data integration methods.

The key finding from this evaluation based on mapping enzyme names (see Table 4) is that the more flexible synonym and StructAlign algorithms, in particular those using the related synonyms variant, show very equivalent precision to the more simple accession mapping methods where all required accessions are present in the data (which was not the case for data used in Table 3). Therefore, these methods can be considered as practical alternatives when direct accession mapping is not possible between two data sources. Furthermore, flexible mapping methods which explore a larger search space among concept names and synonyms have the potential to improve the recall when aligning metabolic networks (see Table 3) without unduly sacrificing precision.

3.3 Visualising Results

The first two data integration steps in ONDEX (Figure 1), i.e. the import of data and subsequent mapping of equivalent and related entities result in large and complex graphs, which may consist of millions of elements. As an illustration, the integration

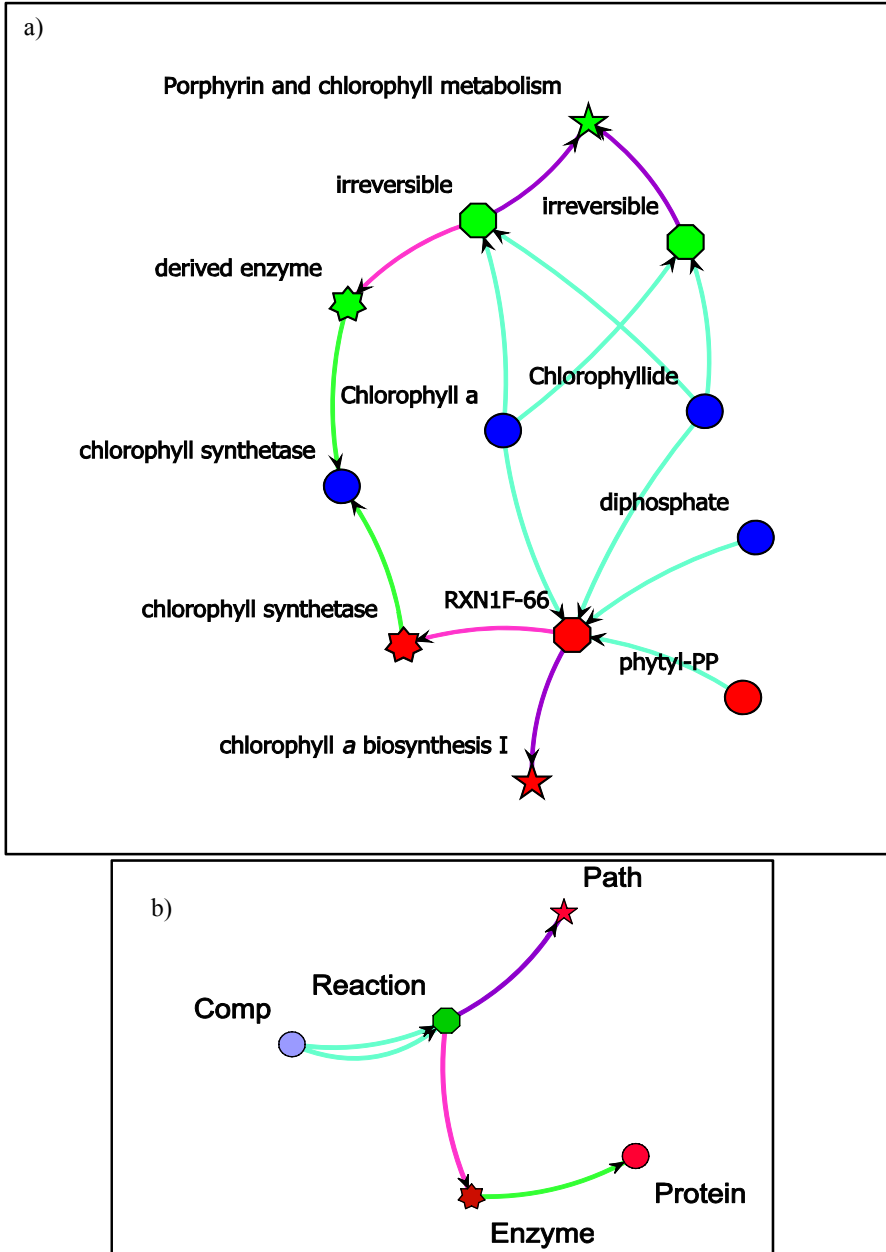


Fig. 3. a) Chlorophyll a biosynthesis I pathway from AraCyc (red nodes) with corresponding subset from KEGG (green nodes). Nodes identified to be equivalent in blue. b) meta-graph providing an overview of the integrated data, node colour and shape distinguishes classes.

of KEGG and AraCyc for this evaluation results in more than 100,000 concepts and relations. The mapping methods were run with optimal parameters identified in Section 3.2. After filtering down to a specific pathway using methods available in the ONDEX visualisation interface, it was possible to extract information from the integrated data presented in Figure 3.

Figure 3a displays parts of the *chlorophyll a biosynthesis I* pathway from AraCyc (lower part of Fig. 3a) mapped to the corresponding subset from KEGG (upper part of Fig. 3a). It is now possible to visualise the differences between the two integrated pathways. AraCyc provides more metabolites (circle, e.g. *phytyl-PP*) on the reaction (octagon, *RXNIF-66*). KEGG denotes also the reverse direction of the reaction *chlorophyllide to chlorophyll a*.

The meta-graph as described in Section 2.3 is shown in Figure 3b. This visualisation shows that the integrated dataset consists of pathways (Path), reactions (Reaction) belonging to these pathways, metabolites (Comp) consumed or produce by the reactions and several combinations of enzymes (Enzyme) and proteins (Proteins) catalysing the reactions. The meta-graph provides the user with a useful high level overview of the conceptual schema for the integrated data.

4 Discussion

In this paper we set out to evaluate alternative methods for creating cross-references (mappings) between information in different but related databases. This is an essential component in the integration of data having different technical and semantic structures. We used two realistic evaluation cases to quantify the performance of a range of different methods for mapping between the concept names and synonyms used in these databases. A quantitative evaluation of these methods shows that a graph based algorithm (StructAlign) and mapping through synonyms can perform as well as using accession codes. In the particular application case of linking chemical compound names between pathway databases, the StructAlign algorithm outperformed the most direct mapping through accession codes by identifying more elements that were indirectly linked. Manual inspection of the false positive mappings showed that both StructAlign and synonym mapping methods can be used where accession codes are not available to provide links between equivalent database concepts. The combination of all three mapping methods yields the most complete projection between different data sources. This is an important result because it is not always possible to find suitable accession code systems that provide the direct cross-references between databases once you move outside the closely related databases that deal with biological sequences and their functional annotations.

A similar approach to StructAlign called “SubTree Match” has been described [24] for aligning ontologies. Our work extends this idea into a more general approach for integrating biological networks and furthermore, presents a formal evaluation in terms of precision and recall, which is missing in [24]. Furthermore, we have shown how semantic constraints on the integrated graph can simplify the presentation of the data by using a meta-graph. This feature, which is of significant value for visualising large integrated datasets, is not available in pure ontology mapping systems like [25] or [26], yet could prove valuable as ontologies become ever larger. While several analysis and verification methods for single pathway databases exist such as presented in

[27], our evaluation has shown the value of being able to compare data sources as a means of identifying conflicting information between related data sources. This is an important quality control check and a prerequisite for detailed comparisons of multiple pathway databases.

A particular challenge in this evaluation has been to identify suitable “gold standard” data sets against which to assess the success of the algorithms we have developed. The results presented here are therefore not definitive, but represent the best quantitative comparison that could be achieved in the circumstances. We believe that the results represent a pragmatic evaluation of the relative performance of the different approaches to concept name matching for data integration of life science database.

The methods presented in this publication are implemented in the ONDEX system.

ONDEX is developed as an Open Source project and can be found at <http://ondex.sourceforge.net/>.

Acknowledgements

Rothamsted Research is supported by the BBSRC. This work is funded by BBSRC Grant BBS/B/13640 & BB/F006039/1. We wish to thank Robert Pesch for his contributions to the AraCyc parser and Stephan Philippi for suggestions including Figure 1.

References

1. Köhler, J., Baumbach, J., Taubert, J., Specht, M., Skusa, A., Rueegg, A., Rawlings, C., Verrier, P., Philippi, S.: Graph-based analysis and visualization of experimental results with ONDEX. *Bioinformatics* 22, 1383–1390 (2006)
2. Gaylord, M., Calley, J., Qiang, H., Su, E.W., Liao, B.: A flexible integration and visualisation system for biomarker discovery. *Applied bioinformatics* 5, 219–223 (2006)
3. Fischer, H.P.: Towards quantitative biology: integration of biological information to elucidate disease pathways and to guide drug discovery. *Biotechnol. Annu. Rev.* 11, 1–68 (2005)
4. Etzold, T., Ulyanov, A., Argos, P.: SRS: information retrieval system for molecular biology data banks. *Methods Enzymol.* 266, 114–128 (1996)
5. Baitaluk, M., Qian, X., Godbole, S., Raval, A., Ray, A., Gupta, A.: PathSys: integrating molecular interaction graphs for systems biology. *BMC bioinformatics* 7, 55 (2006)
6. Küntzer, J., Blum, T., Gerasch, A., Backes, C., Hildebrandt, A., Kaufmann, M., Kohlbacher, O., Lenhof, H.-P.: BN++ - A Biological Information System. *Journal of Integrative Bioinformatics* 3 (2006)
7. Köhler, J., Rawlings, C., Verrier, P., Mitchell, R., Skusa, A., Ruegg, A., Philippi, S.: Linking experimental results, biological networks and sequence analysis methods using Ontologies and Generalized Data Structures. *Silico. Biol.* 5, 33–44 (2004)
8. Shannon, P., Markiel, A., Ozier, O., Baliga, N.S., Wang, J.T., Ramage, D., Amin, N., Schwikowski, B., Ideker, T.: Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.* 13, 2498–2504 (2003)
9. RIKEN: Semantic Web Folders (2009)
10. Köhler, J., Philippi, S., Specht, M., Ruegg, A.: Ontology based text indexing and querying for the semantic web. *Know.-Based Syst.* 19, 744–754 (2006)
11. Bairoch, A.: The ENZYME database in 2000. *Nucleic Acids Res.* 28, 304–305 (2000)

12. Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., Harris, M.A., Hill, D.P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J.C., Richardson, J.E., Ringwald, M., Rubin, G.M., Sherlock, G.: Gene ontology: tool for the unification of biology. *The Gene Ontology Consortium. Nat. Genet.* 25, 25–29 (2000)
13. Kanehisa, M., Goto, S., Kawashima, S., Okuno, Y., Hattori, M.: The KEGG resource for deciphering the genome. *Nucleic Acids Res.* 32, D277–D280 (2004)
14. Mueller, L.A., Zhang, P., Rhee, S.Y.: AraCyc: a biochemical pathway database for Arabidopsis. *Plant Physiol.* 132, 453–460 (2003)
15. Taubert, J., Sieren, K.P., Hindle, M., Hoekman, B., Winnenburg, R., Philippi, S., Rawlings, C., Köhler, J.: The OXL format for the exchange of integrated datasets. *Journal of Integrative Bioinformatics* 4 (2007)
16. Smith, B.: Beyond Concepts: Ontology as Reality Representation. In: Varzi, A., Vieu, L. (eds.) *Proceedings of FOIS* (2004)
17. Baldwin, T.K., Winnenburg, R., Urban, M., Rawlings, C., Köhler, J., Hammond-Kosack, K.E.: PHI-base provides insights into generic and novel themes of pathogenicity. *Molecular Plant-Microbe Interactions* 19, 1451–1462 (2006)
18. Winnenburg, R., Baldwin, T.K., Urban, M., Rawlings, C., Köhler, J., Hammond-Kosack, K.E.: PHI-base: A new database for Pathogen Host Interactions. *Nucleic Acids Res.* 34(Database issue), D459–D464 (2006)
19. Köhler, J., Munn, K., Rüegg, A., Skusa, A., Smith, B.: Quality Control for Terms and Definitions in Ontologies and Taxonomies. *BMC Bioinformatics* 7, 212 (2006)
20. Goutte, C., Gaussier, É.: A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. In: Losada, D.E., Fernández-Luna, J.M. (eds.) *ECIR 2005. LNCS*, vol. 3408, pp. 345–359. Springer, Heidelberg (2005)
21. Green, M.L., Karp, P.D.: The outcomes of pathway database computations depend on pathway ontology. *Nucl. Acids Res.* 34, 3687–3697 (2006)
22. Buntrock, R.E.: Chemical registries—in the fourth decade of service. *J. Chem. Inf. Comput. Sci.* 41, 259–263 (2001)
23. Meinke, D.: Genetic nomenclature guide. *Arabidopsis thaliana. Trends in genetics*, 22–23 (1995)
24. Zhang, L., Gu, J.-G.: Ontology based semantic mapping architecture. In: *Fourth International Conference on Machine Learning and Cybernetics*. IEEE, Los Alamitos (2005)
25. Nov, N.: *The Prompt Tab*, vol. 2008 (2005)
26. Marquet, G., Mosser, J., Burgun, A.: A method exploiting syntactic patterns and the UMLS semantics for aligning biomedical ontologies: The case of OBO disease ontologies. *Int. J. Med. Inform.* 76(suppl. 3), S353–S361 (2007)
27. Racunas, S.A., Shah, N.H., Fedoroff, N.V.: A case study in pathway knowledgebase verification. *BMC bioinformatics* 7, 196 (2006)

Integration of Full-Coverage Probabilistic Functional Networks with Relevance to Specific Biological Processes

Katherine James^{1,2}, Anil Wipat^{1,2}, and Jennifer Hallinan^{1,2}

¹ School of Computing Science, Newcastle University, Newcastle-upon-Tyne,
NE1 7RU, United Kingdom

² Centre for Integrated Biology of Ageing and Nutrition, Ageing Research Laboratories,
Institute for Ageing and Health, Newcastle University,
Campus for Ageing and Vitality, Newcastle-upon-Tyne,
NE4 5PL, United Kingdom

Abstract. Probabilistic functional integrated networks are powerful tools with which to draw inferences from high-throughput data. However, network analyses are generally not tailored to specific biological functions or processes. This problem may be overcome by extracting process-specific sub-networks, but this approach discards useful information and is of limited use in poorly annotated areas of the network. Here we describe an extension to existing integration methods which exploits dataset biases in order to emphasise interactions relevant to specific processes, without loss of data. We apply the method to high-throughput data for the yeast *Saccharomyces cerevisiae*, using Gene Ontology annotations for ageing and telomere maintenance as test processes. The resulting networks perform significantly better than unbiased networks for assigning function to unknown genes, and for clustering to identify important sets of interactions. We conclude that this integration method can be used to enhance network analysis with respect to specific processes of biological interest.

Keywords: Integrated networks, relevance, network analysis, clustering.

1 Introduction

One of the fundamental goals of systems biology is the experimental verification of the interactome: the entire complement of molecular interactions within an organism. This goal requires the systematic confirmation of all interactions occurring between the proteins, genes, ligands and other molecules of the cell. However, even for a relatively simple organism such as the baker's yeast *Saccharomyces cerevisiae*, verification of the interactome is a substantial task, involving thousands of interactions occurring at all stages of the cell cycle, and in response to all possible external stimuli [1]. In multicellular organisms such as humans, the interactome must also encompass all tissue types, ages and disease states [2, 3]. In order to achieve this massive undertaking, various experimental, computational and statistical methods have been developed to determine each protein's function and interactions [4-6] and hundreds of public databases have been created to store this information [7, 8].

Recent advances in high-throughput technology have led to the development of experimental techniques with which protein interactions and functional associations can be studied on a genome/proteome-wide scale [9, 10]. These technologies, such as tandem affinity purification-mass spectroscopy [11], yeast 2-hybrid [12] and fluorescence resonance energy transfer [13], have produced a wealth of information defining the interactions within a cell and predicting protein functions [9]. However, the number of overlapping interactions between datasets from different experiments can be surprisingly low [14-17] and each individual experimental type can only provide information about certain aspects of the cell's behaviour and interactions [18]. For instance, a study by Beyer *et al.* [19] identified several synthetic lethal protein pairs: proteins that can be deleted from the cell individually but when deleted together result in an inviable cell. They found that only 1% of the pairs interact physically, indicating that this experimental technique identifies indirect, rather than direct, associations. In order to fully characterise every aspect of highly complex cellular systems, and infer new knowledge from the data, diverse data sources must be integrated in a systematic fashion [20, 21].

1.1 Network Analysis

One of the most powerful computational approaches to the integration of heterogeneous data and the prediction of protein functions is network analysis, which allows interactions to be visualised and studied on a cell-wide scale using graph theory. Nodes in an interactome graph correspond to genes or gene products and edges represent a summary of the evidence for interaction between them. More complex graph models can include nodes representing other entities, such as pathways, ligands, annotations and publication references [22]. The simplest networks include a connection between two nodes if there is evidence of a functional link between them from at least one data source [23]. The resulting networks can be used for a number of applications: for example, to detect protein complexes [24-26]; to predict protein functions [27, 28]; or to infer novel interactions that were not detected experimentally [29, 30].

The quality of different datasets, in terms of coverage of the genome and accuracy of the identification of interactions, varies with the experimental technique used. Recently, several methods have been employed to assess data quality prior to integration. The most common method involves scoring data against a high-quality, usually manually curated, gold standard dataset [31]. Methods for reliability scoring have included Bayesian data fusion [21, 32], Support Vector Machines [33] and Markov random fields [34, 35]. The scored networks, termed probabilistic functional integrated networks (PFINs), are annotated with edge weights representing confidence scores. This type of network has been used to analyse data from several different species, including yeast [36, 37], mouse [38, 39] and human [40], and to make comparisons across multiple species [41].

1.2 Probabilistic Networks

PFINs, while far more informative than single source and unweighted networks, have several problems. Firstly, the high-throughput data is very noisy, with estimates of false positive rates varying from 20% to as high as 91% depending upon the

technology used to generate the data [1, 42, 43]. Secondly, many studies have shown that the networks as a whole, and the data sources individually, may be biased towards certain cellular processes [14, 44]. For instance, co-expression data shows significant bias towards interactions between genes involved in ribosome biogenesis [45]. Current approaches to these problems usually involve attempting to remove the noise [46, 47] or bias [48]. However, these approaches may further complicate the analysis by removing true positive interactions, resulting in a loss of valid data.

An additional drawback is that existing network integration methods do not take into account the relevance of each experimental data set to specific biological processes, so that functional prediction techniques applied to the networks are global rather than tailored to a specific area of biology. For instance, a study by Karaoz *et al.* [28] produced 208 novel protein annotations ranging across 96 Gene Ontology (GO) [49] biological process terms. While this is a large amount of new data, an individual research group may only be interested in processes involving one or two of these terms. Several studies have approached the issue of process relevance using existing annotation data to either extract process-specific sub-networks from the PFIN [31, 50] or to build process-specific networks using a subset of the data [18, 51]. These methods also discard potentially useful information. Further, they are of limited use in areas of the network that contain a large proportion of unannotated proteins.

1.3 Dataset Relevance

We contend that the biases present in experimental data can be a valuable source of information. Bias exists in data for several reasons: the type of experimental technique or conditions chosen may be suited to the detection of particular type of interaction; the experiment itself may have been designed to research a particular process; or the final data chosen for publication may reflect the topic of the area of interest [52]. The combination of these factors gives each dataset its own unique biases and consequently, when analyzing the data, some datasets will be more informative about a particular biological process than other datasets.

In this report we describe an algorithm which extends existing integration methods to exploit, rather than eliminate, data bias in order to optimise network predictions relevant to specific processes. We define relevance as association to the process of interest (POI). The relevance of a dataset is the proportion of the data with known annotations to the POI. In the case of network predictions, nodes that are predicted to belong to the POI or that cluster with nodes annotated to the POI are considered relevant. The integration technique involves scoring each dataset's relevance to a POI and using this information during integration of the confidence scores from each dataset. Our algorithm can be used to modify any existing scoring schema. The resulting networks can be used to generate new hypotheses regarding the POI using existing network analysis techniques and, unlike process-specific sub-networks, none of the original experimental data is lost.

We applied our algorithm to the study of the role of telomeres in ageing, using existing *Saccharomyces cerevisiae* data. Telomeres are repetitive regions of non-coding DNA that protect the end of the chromosome from degradation [53]. Every time a cell divides the telomeres become shorter. The shortening of the telomeres has been linked with the ageing process, since as telomeres become critically short the cell becomes senescent [54].

We generated networks relevant to the GO annotations Telomere Maintenance (GO:0000723) and Ageing (GO:0007568), and compared them with networks integrated from the same datasets without consideration of any biological process relevance.

Systematic evaluation of three ageing-relevant networks using existing network analysis techniques showed that the relevance networks show a statistically significant improvement over the control network, both in predicting the function of ageing-related nodes and in clustering genes to identify biologically important sets of interactions. This integration technique enhances the results of network analysis for individual research groups with specific interests.

2 Methods

Version 38 of the BioGRID database [55] for *Saccharomyces cerevisiae* was used as the source of interaction data. The protein pairs were first split by PubMed¹ ID to identify pairs produced by different experimental studies. Self-interactions, duplicates and interactions with proteins from other species were removed. Datasets containing at least 100 interactions were treated as separate high-throughput (HTP) data sources while the remaining low-throughput (LTP) studies were grouped together in accordance with the 22 experimental categories provided by BioGRID [56].

2.1 Network Integration

Two scores were produced for each of the datasets: a standard *confidence score* representing its reliability, and a *relevance score* for the POI. The confidence score was calculated by scoring the datasets against a Gold Standard KEGG Pathways [57] dataset using the Bayesian statistics approach developed by Lee *et al.* [21] which calculates a log likelihood score for each dataset (Equation 1).

$$LLS = \ln \left(\frac{P(L|E) / \sim P(L|E)}{P(L) / \sim P(L)} \right). \quad (1)$$

where $P(L|E)$ and $\sim P(L|E)$ represent the frequencies of linkages (L) observed in dataset (E) between genes annotated to the same and differing KEGG pathways respectively, and, $P(L)$ and $\sim P(L)$ represent the prior expectation of linkages between genes in the same and differing KEGG pathways, respectively. Datasets that did not have a positive score were discarded.

For the relevance score, annotation data for *S. cerevisiae* was obtained from the *Saccharomyces* Genome Database [58] and from the Gene Ontology Consortium. The Gene Ontology is a controlled vocabulary, structured as a directed acyclic graph, which describes the molecular function of proteins, the biological processes they are involved in and the cellular compartments in which they are found. A one-tailed Fishers Exact Test was applied to each dataset to score over-representation of genes annotated to the POI, producing the relevance score. The POI was defined as that term and any descendant of that term within the GO graph, with the exclusion of annotations with the evidence code *inferred from electronic annotation* (IEA).

¹ <http://www.ncbi.nlm.nih.gov/pubmed/>

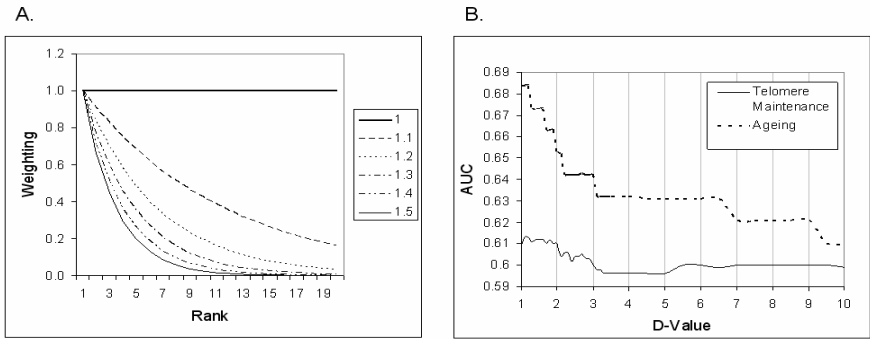


Fig. 1. A. Effect of the D parameter on the weighting given to dataset scores. A value of 1 results in a simple sum while higher weights successively downweights values. B. Effect of the D parameter on the area under curve (AUC) for functional prediction of the two processes. A value of 1.1 was shown to optimise AUC.

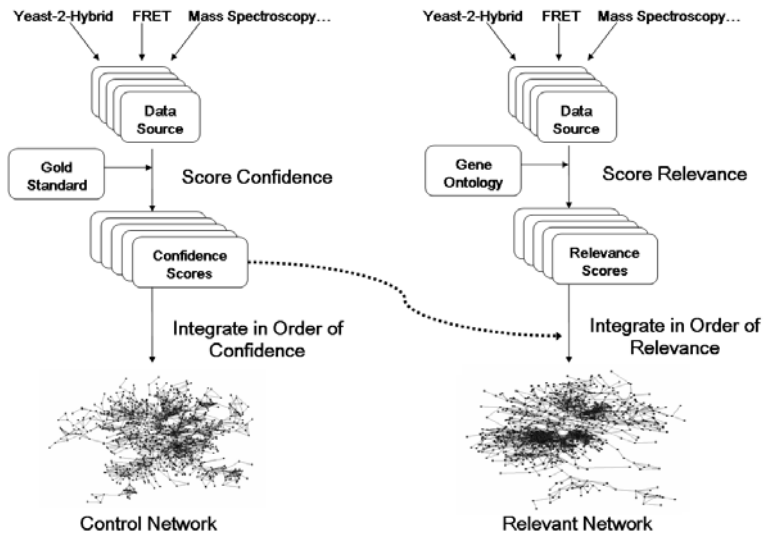


Fig. 2. Overview of the integration method. On the left side a control network is produced by integration of the datasets in order of confidence. On the right, relevance networks are produced by integration of the confidence scores in order of relevance to the process of interest. The two networks produced have the same topology but differ in the edge weightings between nodes.

The control network was produced by integrating the confidence scores using the method (Equation 2) described by Lee *et al.* [21]. This method integrates the datasets in order of their confidence scores, giving a higher weighting to datasets with higher confidence.

$$WS = \sum_{i=1}^n \frac{L_i}{D^{(i-1)}} \quad (2)$$

where L_1 is the highest confidence score and L_n that with the lowest of a set of n datasets. Division of the score by the D parameter means that, while the highest score is integrated unchanged, subsequent weights are progressively downweighted (Fig 1 A).

With a D value of 1 the integration is a simple sum of the scores. A D value of 1.1 was chosen for our purposes, since higher values mean that lower-ranking scores contribute little or nothing to the integration, again discarding potentially important information and reducing network performance (Fig 1 B). In the resulting network highly weighted edges have high confidence.

To produce the relevance networks the relevance scores were used to re-order the datasets prior to integration of the confidence scores, giving a higher weighting to datasets with higher relevance (Fig 2). In the resulting network highly weighted edges have both high confidence and high relevance to the POI.

2.2 Network Evaluation

The networks were evaluated using two previously published network analysis techniques: functional prediction and clustering.

The networks' functional prediction power was evaluated using the maximum weight decision rule described by Linghu [59]. This method assigns annotations to a node based upon the annotations of the first neighbour node with the highest-weighted edge. Leave-one-out cross-validation of known annotations to the POI was carried out using this algorithm for both the relevance and the control networks. The performance of the networks was evaluated using Receiver Operating Characteristic (ROC) curves. In order to compare the relevance and control curves the error of the area under curve (AUC) was calculated using the standard error of the Wilcoxon statistic SE(W) [60, 61].

In the second stage of evaluation, the networks were clustered using the MCL Markov-based clustering algorithm [62]. The MCL algorithm divides the entire network into non-overlapping clusters based on their topology and edge weights. The default inflation value of 1.8 was used in each case. The clusters containing nodes annotated to terms of interest were identified in the relevance and control networks for visual comparison.

3 Results

Division of the BioGRID data produced 72 datasets: 50 HTP and 22 LTP. Dataset size ranged from 14421 interactions ("Collins 17314980 Phenotypic Enhancement") to as few as 33 interactions ("Protein-RNA"). Twenty-seven datasets were lost due to negative scores against the gold standard data. These datasets included many of the smaller HTP datasets and the smallest of the combined LTP datasets (<100 interactions).

Three networks were integrated using the relevance ranking method: a network with relevant to the GO term Ageing; a network relevant to Telomere Maintenance; and one with relevance to both ageing and telomere maintenance produced by combining the terms of interest. The control network was integrated using the confidence

ranked scores. The four networks were topologically identical, having the same number of nodes and edges, with the same connectivity. However, the edge weights differed reflecting the different order of confidence score integration. Table 1 summarises the integration order for the top eight of the 45 datasets.

Table 1. A comparison of integration order for 8 datasets in the relevance and control networks. Datasets were integrated in order from rank 1 to 8.

Dataset	Confidence Score	Confidence Rank	Ageing Rank (A)	Telomere Rank (B)	Combined A & T Rank
Co-Crystal Structure	6.6937	1	4	6	6
Krogan (14759368)	5.7054	2	8	7	8
Collins (17200106)	5.7040	3	6	2	2
Co-Localisation	5.0842	4	3	4	4
Co-Purification	4.9335	5	7	5	5
Co-Fractionation	4.5212	6	5	8	7
Affinity Capture Western	4.4641	7	1	1	1
Two Hybrid	4.2253	8	2	3	3

3.1 Functional Prediction

Leave-one-out functional prediction of terms of interest was carried out for each of the relevance networks the control network. The results were analyzed using ROC curves (Fig. 3). The control network produced an area under curve (AUC) of 0.613,

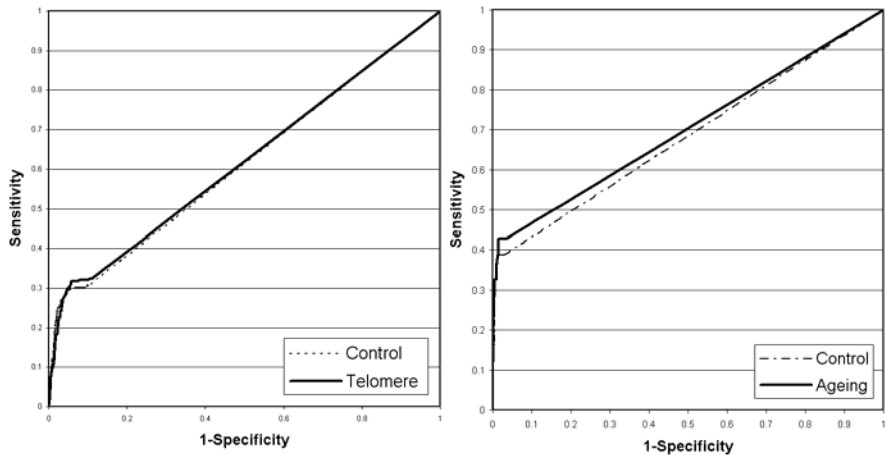


Fig. 3. ROC curves of functional prediction of telomere maintenance and ageing genes for the relevance networks compared with the control

Table 2. Summary of the AUC measurements for the three networks

Process of Interest		Relevant AUC	Relevant SE(W)	Control AUC	Control SE(W)
Telomere Maintenance	Main-	0.618	0.00035	0.613	0.00035
Ageing		0.702	0.00177	0.684	0.00180
Combined		0.640	0.00030	0.635	0.00030

0.684 and 0.640 for ageing, telomere maintenance and the combined terms respectively. The telomere maintenance and combined network AUCs were both improved by 0.005, while the ageing network AUC increased by 0.018. Computation of the standard error of the Wilcoxon statistic, SE(W) showed the improvements were statistically significant in each case (Table 2).

3.2 Clustering

Clustering of the four networks was carried out using the MCL algorithm. The clusters were analysed for the presence of nodes annotated to the POI. The control network produced 573 clusters ranging in size from 164 nodes to 1 node. The relevance networks all produced fewer clusters than the control; the telomere maintenance network had 523 clusters, the ageing network had 508, and combined ageing and telomere maintenance network had 523. The cluster size distribution for all of the four networks was found to be scale free (data not shown).

Clusters of interest (COIs), were defined as clusters containing at least one node of interest. Analysis of the clusters showed that each of the relevance networks clustered into fewer clusters than the control but contained a higher percentage of COIs. The average proportion of COIs in the network increased as the minimum cluster size was raised (Table 3). Similar analysis for a term unrelated to the POI, Maintenance of Protein Location (GO:0045185), showed that the total percentage of COIs was higher for the relevance network in all three cases but that it slowly dropped below the control as cluster size cut-off increased.

Table 3. A summary of the percentage clusters of interest (COI) for the networks in relation to the process of interest. A. Telomere Maintenance. B. Ageing. C. Combined Relevance.

	Network	No. Clusters	Total % COIs	> 2 nodes	> 3 nodes	> 4 nodes
A	Non-Relevant	573	21.29	26.14	28.86	35.19
	Relevant	523	22.37	27.73	31.75	36.92
B	Non Relevant	573	5.06	6.14	7.02	6.53
	Relevant	508	6.50	7.73	8.90	8.59
C	Non Relevant	573	24.26	29.55	33.80	37.98
	Relevant	523	24.67	29.83	33.33	38.35

4 Discussion

Using our extended integration technique we created three relevance networks; one relevant to ageing, one to telomere maintenance, and a combined network with relevance to both terms. The shortening of telomeres over time has been linked to the ageing process and we predicted some overlap between the ageing network and the telomere maintenance network, which we would expect to be enhanced in the combined network. The three networks, as expected, had the same topology as the control but differed in their edge weights due to the order of dataset integration.

In the control network the datasets are ranked in order of confidence score, with the highest score ranked first. In the relevance networks, although the scores integrated are still the confidence scores, the datasets are ranked prior to integration in order of relevance score, with the most relevant first. Therefore, in the control network highly weighted edges have high confidence but in the relevance network the highly weighted edges have both high confidence and high relevance. Table 1 illustrates the difference in rankings between eight of the datasets in the four networks. In the control network the highest-ranking dataset is Co-Crystal Structure, which is integrated without modification. In the telomere maintenance network this dataset is ranked sixth and because of the down-weighting caused by division of its confidence score by D^{i-1} (here 1.1⁵) will contribute almost nothing to the integration. The Affinity Capture Western data, ranked top, will be given highest importance. Affinity Capture Western is also ranked first in the ageing network, which is consistent with the expected overlap between networks relevant to ageing and telomere maintenance. The rank order of datasets in the combined network was far closer to that of the telomere maintenance network than the ageing network, since there are far more genes annotated to Telomere Maintenance (276) than to Ageing (49).

Importantly, while the order of integration differs between the datasets, the final scores themselves are calculated as a weighted sum of dataset confidence scores. Consequently, datasets with high relevance but low confidence will have a high rank, but their contribution to the final edge weight will be based on their confidence score and hence still take into account the reliability of the data. This method of integration is therefore extremely flexible and can be used with any existing confidence scoring scheme and any gold standard dataset. For instance, the 27 datasets lost when scored against KEGG may not be lost if scored against an alternate gold standard. The integration method is also easily applicable to any existing annotation schemes, such as MIPS [63] annotations.

4.1 Network Performance

To evaluate whether the relevance networks provide more information about the processes to which they are relevant we applied two existing network analysis techniques: functional prediction and clustering performance.

Many algorithms have been developed for the prediction of gene function from their connectivity. Different algorithms use different decision rules for the transfer of annotations between nodes. While several complex algorithms exist that incorporate both network topology and edge weights into annotation decisions we chose to use an algorithm based solely upon weights as our networks are topologically identical. However, any

prediction algorithm that utilizes edge weights could be used. The maximum weight rule is a simple, local, edge weight-based rule that has been shown to out-perform other local decision rules in functional prediction [59]. The algorithm performed statistically significantly better on the three relevance networks than on the control network. Integrating datasets in order of relevance to the POI does appear to increase the prediction accuracy of this algorithm. The extent to which performance is improved varies with the POI chosen (data not shown). Further research into why this occurs should lead to an understanding of which types of GO annotation are most suited for the relevance-biased approach to integration, and this is an ongoing investigation in our laboratory.

Functional prediction is a purely computational technique and many biologists wish to explore networks visually in order to generate hypotheses relating to their POI. However, full interactome networks are too large to analyse successfully in this way. Network clustering provides a way of breaking the network down into subnetworks that can be visualized easily. We clustered the networks using the MCL Markov-based algorithm. This algorithm was chosen because, unlike many topology-based algorithms, it utilises the edge weights of the network.

The relevance networks produced fewer clusters than the control network. Genes of interest tended to appear in larger clusters in the relevance networks, together with genes with a range of annotations, including unknowns. Genes which in the control network appear in separate clusters, co-occur in the relevance networks making their relationship to each other and to other genes easier to observe and investigate.

Ageing is a complex, multi-factorial, systems-wide phenomenon [64]. Although we constructed our networks to be relevant to a specific GO annotation of interest, we were interested to observe that other ageing-relevant interactions were also over-represented in COIs of the relevance networks, as compared with the control network.

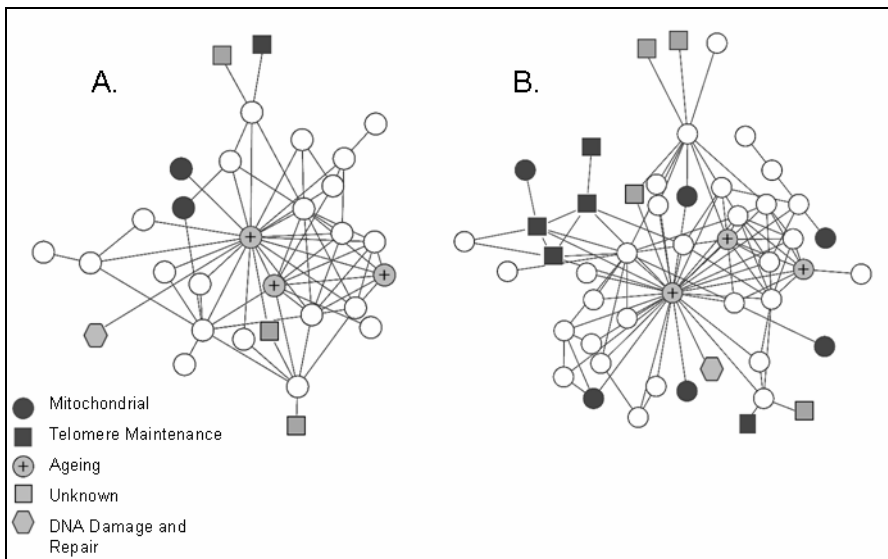


Fig. 4. Example of the clustering of three ageing genes in the control network (A) and in the relevance network (B)

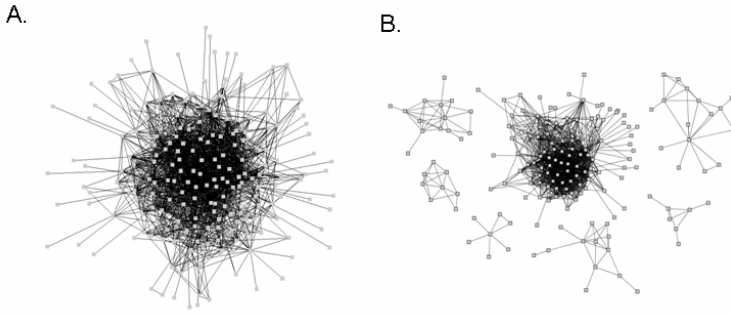


Fig. 5. Overview of the largest telomere maintenance cluster and equivalent clusters in the control network. A. Whole relevant cluster. B. Seven equivalent control clusters.

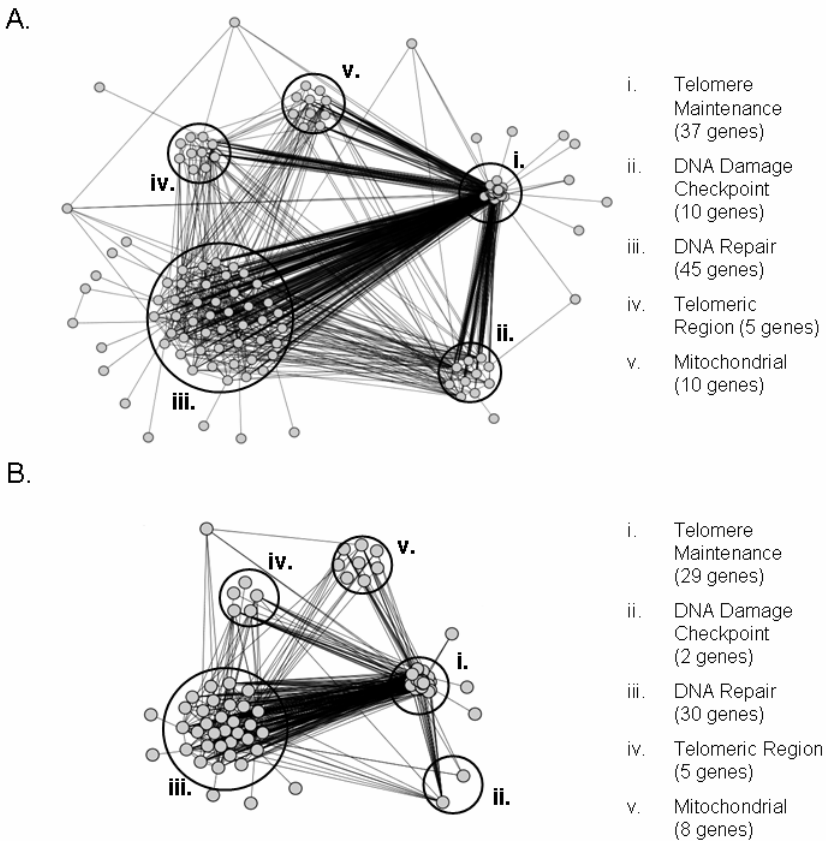


Fig. 6. The clusters from Fig. 5 displaying nodes annotated to ageing-associated processes and unknowns. A. Relevant cluster. B. Largest control cluster.

Interactions involving mitochondria, DNA damage checkpoints, and additional ageing/telomere-associated processes appeared. In the ageing network the genes of interest cluster in the same small groups as in the control network; however, the clusters contain a greater number of additional nodes. An example is shown in Fig 4. A cluster of three ageing genes occurs in the control network, together with four genes annotated to the ageing-associated processes and three genes of unknown function (4A). In the relevance network (4B), the same three genes have clustered with ten genes annotated to the associated processes including four additional telomere maintenance genes and an additional unknown.

In the telomere maintenance relevance network several clusters contained a large number of genes of interest. The largest of these clusters contained 37 genes annotated to telomere maintenance (Fig 5 A). Of these genes 29 were clustered together in the control network while the remainder were spread between six other clusters (B). When broken down into groups of associated genes the relevance network cluster (Fig 6 A) contains far more genes of interest and a greater number of unknowns than the large control cluster (B). The additional information provided by the relevance network clusters provides greater scope to draw inferences from the data and demonstrates the effect of the relevance integration technique in emphasising relevant information within the network.

5 Conclusions

We describe here an algorithm for extending existing data integration techniques to incorporate relevance to process of interest, as identified by Gene Ontology annotations. We find that the relevance networks have the same topology as control networks integrated without consideration of their relevance, but different edge weightings, with interactions between genes relevant to the process of interest weighted up. Analysis of the networks' performance at function prediction shows that they perform significantly better than the control, having a higher proportion of true positive to false positive predictions over a range of thresholds. Cluster analysis using the MCL algorithm generates larger clusters containing more genes of interest in the relevance network, making visual inspection both easier and more informative. Our algorithm retains as much information as possible from the original datasets, whilst highlighting areas of the integrated networks likely to be of interest to researchers with a specific process of interest, and as such is a valuable extension to standard data integration techniques.

Acknowledgments

This work was supported by the BBSRC CISBAN and ONDEX grants (BB/C008200/1 & BB/F529038/1) and by the Research Councils UK.

References

1. Cusick, M.E., Klitgord, N., Vidal, M., Hill, D.E.: Interactome: Gateway into Systems Biology. *Hum. Mol. Genet.* 14(2), 171–181 (2005)
2. Adourian, A., Jennings, E., Balasubramanian, R., Hines, W.M., Damian, D., Plasterer, T.N., Clish, C.B., Stroobant, P., McBurney, R., Verheij, E.R., Bobeldijk, I., van der Greef, J., Lindberg, J., Kenne, K., Andersson, U., Hellmold, H., Nilsson, K., Salter, H., Schuppe-Koistinen, I.: Correlation Network Analysis for Data Integration and Biomarker Selection. *Mol. Biosyst.* 4, 249–259 (2008)
3. Li, C., Li, H.: Network-Constrained Regularization and Variable Selection for Analysis of Genomic Data. *Bioinformatics* 24, 1175–1182 (2008)
4. Godzik, A., Jambon, M., Friedberg, I.: Computational Protein Function Prediction: Are We Making Progress? *Cell Mol. Life Sci.* 64, 2505–2511 (2007)
5. Mellor, J.C., Yanai, I., Clodfelter, K.H., Mintseris, J., DeLisi, C.: Predictome: A Database of Putative Functional Links between Proteins. *Nucleic Acids Res.* 30, 306–309 (2002)
6. von Mering, C., Jensen, L.J., Kuhn, M., Chaffron, S., Doerks, T., Krüger, B., Snel, B., Bork, P.: String 7—Recent Developments in the Integration and Prediction of Protein Interactions. *Nucleic Acids Res.* 35, 358–362 (2007)
7. De Las Rivas, J., de Luis, A.: Interactome Data and Databases: Different Types of Protein Interaction. *Comp. Funct. Genomics.* 5, 173–178 (2004)
8. Galperin, M.Y.: The Molecular Biology Database Collection: 2008 Update. *Nucleic Acids Res.* 36, 2–4 (2008)
9. Marcotte, E., Date, S.: Exploiting Big Biology: Integrating Large-Scale Biological Data for Function Inference. *Brief. Bioinform.* 2, 363–374 (2001)
10. Mathivanan, S., Periaswamy, B., Gandhi, T.K.B., Kandasamy, K., Suresh, S., Mohmood, R., Ramachandra, Y.L., Pandey, A.: An Evaluation of Human Protein-Protein Interaction Data in the Public Domain. *BMC Bioinformatics* 7(suppl. 5) (2006)
11. Rigaut, G., Shevchenko, A., Rutz, B., Wilm, M., Mann, M., Seraphin, B.: A Generic Protein Purification Method for Protein Complex Characterization and Proteome Exploration. *Nat. Biotechnol.* 17, 1030–1032 (1999)
12. Fields, S., Song, O.: A Novel Genetic System to Detect Protein-Protein Interactions. *Nature* 340, 245–246 (1989)
13. Kaganman, I.: Fretting for a More Detailed Interactome. *Nat. Methods* 4, 112–113 (2007)
14. Bader, G.D., Hogue, C.W.V.: Analyzing Yeast Protein-Protein Interaction Data Obtained from Different Sources. *Nat. Biotechnol.* 20, 991–997 (2002)
15. Collins, S.R., Kemmeren, P., Zhao, X.-C., Greenblatt, J.F., Spencer, F., Holstege, F.C.P., Weissman, J.S., Krogan, N.J.: Toward a Comprehensive Atlas of the Physical Interactome of *Saccharomyces Cerevisiae*. *Mol. Cell Proteomics.* 6, 439–450 (2007)
16. Futschik, M.E., Chaurasia, G., Herzel, H.: Comparison of Human Protein-Protein Interaction Maps. *Bioinformatics* 23, 605–611 (2007)
17. Hart, G.T., Lee, I., Marcotte, E.R.: A High-Accuracy Consensus Map of Yeast Protein Complexes Reveals Modular Nature of Gene Essentiality. *BMC Bioinformatics* 8, 236 (2007)
18. Huttenhower, C., Troyanskaya, O.G.: Assessing the Functional Structure of Genomic Data. *Bioinformatics* 24, 330–338 (2008)
19. Beyer, A., Bandyopadhyay, S., Ideker, T.: Integrating Physical and Genetic Maps: From Genomes to Interaction Networks. *Nat. Rev. Genet.* 8, 699–710 (2007)

20. Hallinan, J.S., Wipat, A.: Motifs and Modules in Fractured Functional Yeast Networks. In: IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology (CIBCB 2007), pp. 189–196 (2007)
21. Lee, I., Date, S.V., Adai, A.T., Marcotte, E.M.: A Probabilistic Functional Network of Yeast Genes. *Science* 306, 1555–1558 (2004)
22. Koehler, J., Baumbach, J., Taubert, J., Specht, M., Skusa, A., Rüegg, A., Rawlings, C., Verrier, P., Philippi, S.: Graph-Based Analysis and Visualization of Experimental Results with Ondex. *Bioinformatics* 22, 1383–1390 (2006)
23. Liu, Y., Kim, I., Zhao, H.: Protein Interaction Predictions from Diverse Sources. *Drug Discov. Today* 13, 409–416 (2008)
24. Asthana, S., King, O.D., Gibbons, F.D., Roth, F.P.: Predicting Protein Complex Membership Using Probabilistic Network Reliability. *Genome Res.* 14, 1170–1175 (2004)
25. Bader, G.D., Hogue, C.W.V.: An Automated Method for Finding Molecular Complexes in Large Protein Interaction Networks. *BMC Bioinformatics* 4, 2 (2003)
26. Brun, C., Herrmann, C., Guenoche, A.: Clustering Proteins from Interaction Networks for the Prediction of Cellular Functions. *BMC Bioinformatics* 5, 95 (2004)
27. Chua, H.N., Sung, W.-K., Wong, L.: Using Indirect Protein Interactions for the Prediction of Gene Ontology Functions. *BMC Bioinformatics* 8(suppl. 4) (2007)
28. Karaoz, U., Murali, T.M., Letovsky, S., Zheng, Y., Ding, C., Cantor, C.R., Kasif, S.: Whole-Genome Annotation by Using Evidence Integration in Functional-Linkage Networks. *Proc. Natl. Acad. Sci. U. S. A.* 101, 2888–2893 (2004)
29. Clauset, A., Moore, C., Newman, M.E.J.: Hierarchical Structure and the Prediction of Missing Links in Networks. *Nature* 453, 98–101 (2008)
30. Gilchrist, M.A., Salter, L.A., Wagner, A.: A Statistical Framework for Combining and Interpreting Proteomic Datasets. *Bioinformatics* 20, 689–700 (2004)
31. Myers, C.L., Troyanskaya, O.G.: Context-Sensitive Data Integration and Prediction of Biological Networks. *Bioinformatics* 23, 2322–2330 (2007)
32. Li, J., Li, X., Su, H., Chen, H., Galbraith, D.W.: A Framework of Integrating Gene Relations from Heterogeneous Data Sources: An Experiment on *Arabidopsis Thaliana*. *Bioinformatics* 22, 2037–2043 (2006)
33. Yellaboina, S., Goyal, K., Mande, S.C.: Inferring Genome-Wide Functional Linkages in *E. Coli* by Combining Improved Genome Context Methods: Comparison with High-Throughput Experimental Data. *Genome Res.* 17, 527–535 (2007)
34. Deng, M., Chen, T., Sun, F.: An Integrated Probabilistic Model for Functional Prediction of Proteins. *J. Comput. Biol.* 11, 463–475 (2004)
35. Jaimovich, A., Elidan, G., Margalit, H., Friedman, N.: Towards an Integrated Protein-Protein Interaction Network: A Relational Markov Network Approach. *J. Comput. Biol.* 13, 145–164 (2006)
36. Chen, Y., Xu, D.: Global Protein Function Annotation through Mining Genome-Scale Data in Yeast *Saccharomyces Cerevisiae*. *Nucleic Acids Res.* 32, 6414–6424 (2004)
37. Kiemer, L., Costa, S., Ueffing, M., Cesareni, G.: Wi-Phi: A Weighted Yeast Interactome Enriched for Direct Physical Interactions. *Proteomics* 7, 932–943 (2007)
38. Guan, Y., Myers, C.L., Lu, R., Lemischka, I.R., Bult, C.J., Troyanskaya, O.G.: A Genomewide Functional Network for the Laboratory Mouse. *PLoS Comput. Biol.* 4 (2008)
39. Kim, W.K., Krumpelman, C., Marcotte, E.M.: Inferring Mouse Gene Functions from Genomic-Scale Data Using a Combined Functional Network/Classification Strategy. *Genome Biol.* 9(suppl. 1) (2008)
40. Kann, M.G.: Protein Interactions and Disease: Computational Approaches to Uncover the Etiology of Diseases. *Brief Bioinform.* 8, 333–346 (2007)

41. Geisler-Lee, J., O'Toole, N., Ammar, R., Provart, N.J., Millar, A.H., Geisler, M.: A Predicted Interactome for *Arabidopsis*. *Plant Physiol.* 145, 317–329 (2007)
42. Lin, X., Liu, M., Chen, X.-w.: Protein-Protein Interaction Prediction and Assessment from Model Organisms. In: *BIBM 2008: Proceedings of the 2008 IEEE International Conference on Bioinformatics and Biomedicine*, pp. 187–192 (2008)
43. Mrowka, R., Patzak, A., Herzel, H.: Is There a Bias in Proteome Research? *Genome Res.* 11, 1971–1973 (2001)
44. Tanay, A., Sharan, R., Kupiec, M., Shamir, R.: Revealing Modularity and Organization in the Yeast Molecular Network by Integrated Analysis of Highly Heterogeneous Genomewide Data. *Proc. Natl. Acad. Sci. U. S. A.* 101, 2981–2986 (2004)
45. Myers, C.L., Barrett, D.R., Hibbs, M.A., Huttenhower, C., Troyanskaya, O.G.: Finding Function: Evaluation Methods for Functional Genomic Data. *BMC Genomics* 7, 187 (2006)
46. Chen, J., Hsu, W., Lee, M.L., Ng, S.-K.: Discovering Reliable Protein Interactions from High-Throughput Experimental Data Using Network Topology. *Artif. Intell. Med.* 35, 37–47 (2005)
47. Chen, J., Hsu, W., Lee, M.L., Ng, S.-K.: Increasing Confidence of Protein Interactomes Using Network Topological Metrics. *Bioinformatics* 22, 1998–2004 (2006)
48. Lee, I., Li, Z., Marcotte, E.M.: An Improved, Bias-Reduced Probabilistic Functional Gene Network of Baker's Yeast, *Saccharomyces Cerevisiae*. *PLoS ONE* 2 (2007)
49. Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., Harris, M.A., Hill, D.P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J.C., Richardson, J.E., Ringwald, M., Rubin, G.M., Sherlock, G.: Gene Ontology: Tool for the Unification of Biology. The Gene Ontology Consortium. *Nat. Genet.* 25, 25–29 (2000)
50. Guo, Z., Li, Y., Gong, X., Yao, C., Ma, W., Wang, D., Li, Y., Zhu, J., Zhang, M., Yang, D., Wang, J.: Edge-Based Scoring and Searching Method for Identifying Condition-Responsive Protein-Protein Interaction Sub-Network. *Bioinformatics* 23, 2121–2128 (2007)
51. Li, Y., Ma, W., Guo, Z., Yang, D., Wang, D., Zhang, M., Zhu, J., Li, Y.: Characterizing Proteins with Finer Functions: A Case Study for Translational Functions of Yeast Proteins. In: *Bioinformatics and Biomedical Engineering, 2007. ICBBE 2007* pp. 141–144 (2007)
52. Wodak, S.J., Pu, S., Vlasblom, J., Seraphin, B.: Challenges and Rewards of Interaction Proteomics. *Mol. Cell Proteomics* 8, 3–18 (2009)
53. Blackburn, E.H.: Switching and Signaling at the Telomere. *Cell* 106, 661–673 (2001)
54. Sozou, P.D., Kirkwood, T.B.: A Stochastic Model of Cell Replicative Senescence Based on Telomere Shortening, Oxidative Stress, and Somatic Mutations in Nuclear and Mitochondrial DNA. *J. Theor. Biol.* 213, 573–586 (2001)
55. Stark, C., Breitkreutz, B.-J., Reguly, T., Boucher, L., Breitkreutz, A., Tyers, M.: Biogrid: A General Repository for Interaction Datasets. *Nucleic Acids Res.* 34, 535–539 (2006)
56. Reguly, T., Breitkreutz, A., Boucher, L., Breitkreutz, B.-J., Hon, G.C., Myers, C.L., Parsons, A., Friesen, H., Oughtred, R., Tong, A., Stark, C., Ho, Y., Botstein, D., Andrews, B., Boone, C., Troyanskaya, O.G., Ideker, T., Dolinski, K., Batada, N.N., Tyers, M.: Comprehensive Curation and Analysis of Global Interaction Networks in *Saccharomyces Cerevisiae*. *J. Biol.* 5, 11 (2006)
57. Kanehisa, M., Goto, S.: KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res.* 28, 27–30 (2000)

58. Dwight, S.S., Harris, M.A., Dolinski, K., Ball, C.A., Binkley, G., Christie, K.R., Fisk, D.G., Issel-Tarver, L., Schroeder, M., Sherlock, G., Sethuraman, A., Weng, S., Botstein, D., Cherry, J.M.: *Saccharomyces* Genome Database (SGD) Provides Secondary Gene Annotation Using the Gene Ontology (GO). *Nucleic Acids Res.* 30, 69–72 (2002)
59. Linghu, B., Snitkin, E.S., Holloway, D.T., Gustafson, A.M., Xia, Y., DeLisi, C.: High-Precision High-Coverage Functional Inference from Integrated Data Sources. *BMC Bioinformatics* 9, 119 (2008)
60. Hanley, J.A., McNeil, B.J.: The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology* 143, 29–36 (1982)
61. Henderson, A.R.: Assessing Test Accuracy and Its Clinical Consequences: A Primer for Receiver Operating Characteristic Curve Analysis. *Ann. Clin. Biochem.* 30(Pt 6), 521–539 (1993)
62. Enright, A.J., Van Dongen, S., Ouzounis, C.A.: An Efficient Algorithm for Large-Scale Detection of Protein Families. *Nucleic Acids Res.* 30, 1575–1584 (2002)
63. Tetko, I.V., Brauner, B., Dunger-Kaltenbach, I., Frishman, G., Montrone, C., Fobo, G., Ruepp, A., Antonov, A.V., Surmeli, D., Mewes, H.-W.: MIPS Bacterial Genomes Functional Annotation Benchmark Dataset. *Bioinformatics* 21, 2520–2521 (2005)
64. Kirkwood, T.: Ageing: Too Fast by Mistake. *Nature* 444, 1015–1017 (2006)

OpenFlyData: The Way to Go for Biological Data Integration

Jun Zhao, Alistair Miles, Graham Klyne, and David Shotton

Department of Zoology, University of Oxford
South Parks Road, Oxford, OX3 1PS, UK

{jun.zhao, alistair.miles, graham.klyne, david.shotton}@zoo.ox.ac.uk

Abstract. Although many applications have attempted to provide integrative access to distributed datasets, domain scientists largely continue to harvest research data in a conventional way, by consulting each resource independently, because such applications do not fully meet users' needs. This paper describes OpenFlyData (<http://openflydata.org/>), a simple user-led service providing *Drosophila* researchers with integrated access to distributed information. This is an exemplar lightweight solution to the problems of data integration, in which accurate and explicit data modelling enables high precision information retrieval.

Keywords: data integration, user-led, life science, semantic web.

1 Introduction

Bringing data together from different data sources is a necessary yet time-consuming activity for many areas of biology, not least the developmental biology of the fruit fly (*Drosophila melanogaster*). Where and when a given gene is expressed in the organism provide important clues as to the possible role of that gene in development. For each gene, there may be many types of evidence which need to be combined to give an overall picture from which insights can be drawn. For *Drosophila*, these include data from microarray experiments, such as those stored in the FlyAtlas database [4]; and images and annotations from *in situ* hybridisation experiments, such as those in the Berkeley *Drosophila* Genome Project (BDGP) database for *Drosophila* embryos [14] and the FlyTED database for *Drosophila* testis [2,7]; and FlyBase [15], the central *Drosophila* genetics database which provides trustworthy central information resource.

Several projects, such as FlyMine [8], provide one-stop access to diverse life science datasets through a centralized data warehouse approach. Customised specifically for data on *Drosophila* and *Anopheles*, FlyMine provides a collection of easy-to-use template queries and a powerful query builder facility, allowing users to construct arbitrary queries over the FlyMine data model. However, this query builder interface tends to be too generic to use for many biologists without an informatics background and it is impossible to curate and publish all experimental data in one such central location. Alternative data linking approaches reduce the effort required for maintaining a central data repository [6,12,13].

Examples include Bio2RDF [1], workflow approaches such as Taverna [9] and BioMoby [17], and the Semantic Web Pipes project [10].

In the FlyWeb Project [1], we have adopted elements of the above approaches to create OpenFlyData [2], developing a simple Web interface for data integration and a set of lightweight services that could be reused to create new applications.

2 Building the OpenFlyData Software System

We built a Web-based system based on a loosely coupled architecture. This enabled us to use existing tools, to combine tools developed for simple tasks to build newer, more complex applications, and to switch software components given changing needs. We used Semantic Web standards and technologies to promote interoperability of the data resources and tools we publish and to lower the cost of development. To deploy a flexible data model across distributed data resources, we used the Resource Description Framework (RDF) to represent our dataset and Unique Resource Identifiers (URI) to identify each unique resource. To provide a standard, programmatic access to our data sources, we publish our RDF datasets through SPARQL endpoints, allowing data to be programmatically accessed via SPARQL [5][11], and used OWL to represent data schemas.

2.1 System Architecture

We use the Web as our platform, which is an environment that is familiar to biologists and can be accessed using a standard Web browser from any operating system. The client browser renders HTML pages that provide the main interface, while JavaScript and AJAX provide user interactivity, through a *Drosophila*-specific user interface ‘FlyUI’ [3] (see Figure 1), implemented using the Yahoo! User Interface Library YUI [4].

We build this rich AJAX client using a widget pattern. A widget implements a specific task, e.g., retrieving information from a data source or mapping different gene names. The main functionalities of a widget are to receive users’ interaction events and control responses to these events, such as initialising a query to a data source in response to a query string entered by the user, organizing query results as a data model object when being notified the completion of the query, or presenting the result data model to the browser when query completes. Information is retrieved from each data source using SPARQL queries. The coordination of widget(s) is controlled by an application (see Figure 1).

To provide SPARQL access to each data source, we used an off-the-shelf Jena TDB RDF store [5] and its SPARQL protocol implementation Joseki [6] for initial

¹ <http://flyweb.info>

² <http://openflydata.org>

³ <http://code.google.com/p/flyui/>

⁴ <http://developer.yahoo.com/yui/>

⁵ <http://jena.sourceforge.net/TDB/>

⁶ <http://www.joseki.org/>

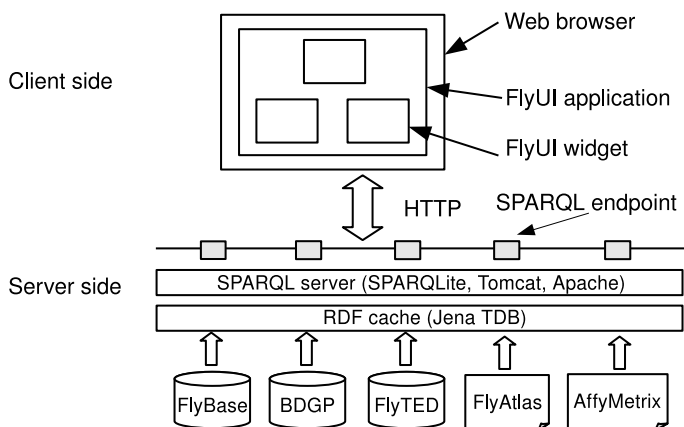


Fig. 1. Schematic view of the OpenFlyData architecture

prototypes. However, because Joseki does not support fine-grained control over the SPARQL service, we built SPARQLite⁷, a simple SPARQL servlet, to provide more configurable and robust public SPARQL services. SPARQLite enables us to make our data sources programmatically available through HTTP/Restful SPARQL endpoints that receive HTTP GET or POST requests containing the SPARQL queries to be evaluated, and return results in JSON (Javascript Object Notation) or RDF/XML formats. At the time of writing, to maintain quality of service, we limit the capability of SPARQLite: We place a ceiling on the number of results returned for each query; we return results in JSON format only for SELECT and ASK queries and in RDF/XML format only for CONSTRUCT and DESCRIBE queries; we disallow queries containing variable predicates, “filter” clauses and unbound variables; and we limit the maximum number of requests sent from any one source to five in any one second.

2.2 Data Management

The data resources integrated by the OpenFlyData applications are FlyBase, FlyTED, BDGP and FlyAtlas, which are highly heterogeneous. FlyBase is used solely to provide the gene name information that enables us to link together the three gene expression data resources.

To provide SPARQL endpoints for these heterogeneous data sources, there were two main options. We could either (i) convert each source data to RDF, then cache the data into an RDF store that can be accessed using a SPARQL query engine; or (ii) re-write SPARQL queries into the native query language of the source data (e.g. SQL), enabling them to be evaluated against the data in their native form. The latter approach is more desirable if the source data have a high churn rate. However, experience with query rewriting over a copy of the BDGP relational database using the D2R server [3] showed that this approach

⁷ <http://code.google.com/p/sparqlite/>

did not scale well if the query result was large. Since the update rate of our data sources is sufficiently low (\sim monthly), we opted to set up native RDF stores for all the datasets, transforming data from the four sources into RDF and making them accessible via four SPARQL endpoints (Figure II).

To generate RDF data from the relational FlyBase and BDGP databases, we used the D2RQ relational-to-RDF mapping language⁸ and D2R toolkits to generate the RDF data in N-Triples³. The FlyBase gene nomenclature dataset comprises approx. 10 million triples, and no scalability problems were encountered, the dump executing in approximately 15 minutes. For FlyAtlas, whose data can be downloaded as an Excel spreadsheet, we wrote a small Python script to convert this spreadsheet file and the related Affymetrix’s annotation table⁹ (as a CSV file) into N-Triple format. For the relational FlyTED database, we adapted software previously developed¹⁸ to convert OAI-PMH¹⁶ (a protocol for metadata harvesting) output to an RDF dump. We chose to represent the original data model for each resource as an OWL ontology, so that we could capture our understanding about each source data through explicit, re-usable documentation in a common language. All the scripts, RDF data and OWL schemas are available at <http://openflydata.org>.

3 Semantic Interoperability

It is important that a scientific data retrieval application provides high precision and recall. Missing or invalid results may lead to false conclusions, and can be costly in wasted time and effort. Therefore, it was essential to provide accurate mappings between the data identifiers used by different data sources.

For *Drosophila* data, FlyBase provides a common point of reference for gene names. It defines a unique symbol (e.g. “schuy”), full name (e.g. “schumacherlevy”) and annotation symbol (e.g. “CG17736”) for each gene. It also provide its own unique FlyBase identifier for each gene which is independent of the other names, e.g. “FBgn0036925”. In addition, FlyBase curates a list of all synonyms found for each gene in the published literature.

Both BDGP and FlyAtlas had already established mapping from their gene names to those used in FlyBase. BDGP associates each of the gene products localized in *in situ* experiments with gene annotation symbols and synonyms imported from FlyBase, including the FlyBase gene identifiers (“FBgn...”). FlyAtlas uses Affymetrix microarray probe identifiers (e.g. “1616608_a_at”) to identify its genes; the mapping of these identifiers to FlyBase gene identifiers (e.g. “FBgn0001128”) is published by Affymetrix as an annotation table specific for each particular Affymetrix microarray.

The mapping from the names used in FlyTED to FlyBase gene identifiers was achieved as part of the FlyWeb project. We first automatically produced a mapping of the 833 probe names used in FlyTED to FlyBase ids, by querying FlyBase using the FlyTED names. We analysed the mapping result and identified

⁸ <http://www.wiwiss.fu-berlin.de/suhl/bizer/d2rq/spec/>

⁹ http://flyatlas.org/drosophila_2.na23.annot.csv

67 probes mapped to no or more than one FlyBase ids. We presented these results to the FlyTED biologists, who manually corrected the ambiguous names. Only one probe remained being mapped to no flybase gene. We finally stored these relationships as part of the FlyTED RDF dataset.

To evaluate the gene name mapping, we tested the accuracy of the SPARQL queries to the FlyTED RDF store and the FlyTED gene expression search service. To test the SPARQL queries to the FlyTED store, we created three sets of test cases to test that the 44 probes with no mapping FlyBase genes are now mapped to correct FlyBase ids; that the 22 probes mapped to more than one FlyBase genes are now uniquely mapped to one FlyBase gene; and that the 17 pairs of FlyTED probes are now uniquely mapped to one FlyBase id after the typos and outdated gene names were fixed. To test the service, we created test data for four different genes to query for FlyTED *in situ* images by any name of these genes and the service returned accurate FlyBase gene ids, FlyTED probe names and images for these genes.

4 The OpenFlyData Application

By the time of writing, the main user-facing application we developed¹⁰, provides the ability to cross search gene expression information using a *Drosophila* gene name or synonym. This application first resolves any ambiguity in the gene name, and then retrieves three different types of evidence regarding the spatial (i.e. the expression location) and temporal (e.g. the development stages during embryogenesis) expression of a given gene (Fig. 2).

The gene name is first disambiguated using the FlyBase RDF dataset:

- If the search gene name matches only a single FlyBase gene, for example, “schuy” matching only “FBgn0036925”, then “FBgn0036925” is used automatically to trigger a search for microarray expression data from FlyAtlas and for images of *in situ* hybridisation from BDGP and FlyTED.
- If the search gene name is ambiguous, matching more than one gene in FlyBase, then all possible matching gene names are displayed in a list, and the user can select one of the possible gene matches. Selecting a gene name will then trigger a search for data from FlyAtlas, BDGP and FlyTED relating only to that gene, thus ensuring both high precision and high recall of results.

Once a gene name has been selected, three queries are sent concurrently to the three additional data sources to retrieve quantitative gene expression information and images relating to that gene:

- Using the Affymetrix mapping data within our FlyAtlas RDF store, probes matching the selected gene are found. The FlyAtlas expression data for the matching probes are retrieved, and presented as a table in the left of Figure 2, showing mRNA expression levels in different tissues of adult *D. melanogaster* and linking each matching probe to its full description at www.flyatlas.org.

¹⁰ <http://openflydata.org/flyui-20081210-FM2-RC4/build/apps/expressionmashup/>

Fig. 2. Cross search for *Drosophila* gene expression information

- Each FlyTED matching image depicting gene expression in *Drosophila* testis is displayed as a thumbnail, with a small caption indicating which strain (genotype) the image depicts, and with a link back to the corresponding full-size image and metadata record at www.fly-ted.org.
- Each image from BDGP depicting gene expression in *Drosophila* embryos is displayed as a thumbnail, with a link to the full size image. The images are grouped by stage of embryogenesis, as they are at <http://www.fruitfly.org/>, and expression pattern annotations are also displayed.

Our intension is to build user-facing applications that are as intuitive and easy to use as possible, being self-explanatory and requiring no initial training or specialist technical knowledge. In the gene expression mashup, three types of data for the selected gene can be retrieved and displayed side-by-side, facilitating quick comparison and validation of the gene expression information about a gene. Users are not required or expected to write any SPARQL queries, a distinctive feature compared with many other semantic web data integration applications.

This relatively simple mashup was chosen as the first application of OpenFlyData because it was most urgently needed by our biologists collaborators and its straightforward requirement enabled us to focus on solving the technical challenges (such as the gene name mapping and the quality of SPARQL service) for building a user-facing semantic web application.

The application in Figure 2 demonstrates the feasibility of combining our lightweight services to create new applications: the full application has been

built by combining four specific widgets, each undertaking a single task. Furthermore, since the release of the gene expression mashup, we have developed new applications, building upon these four widgets, allowing scientists to batch search gene expression information from the three data sources and to retrieve a collection of genes sharing the same gene expression profile and their expression information^[1]. This further proves the feasibility of re-using such lightweight services in alternative ways for building presently unforeseen applications.

5 Conclusion

We have presented our approach for building OpenFlyData, a simple cross-database search service for *Drosophila* gene expression information, and have demonstrated the feasibility of supporting life science data integration using such a loosely coupled software architecture and off-the-shelf Semantic Web tools.

Our user-led approach with rapid prototyping guaranteed that we built an application addressing real user needs. The high precision achieved for our gene name mapping could not have been achieved without our close working relationship with *Drosophila* researchers. The *Drosophila* researchers to whom we have demonstrated OpenFlyData have very much liked the simple and straightforward user interface, and have been helpful in suggesting new functionalities that we might implement, including aggregating results for genes having similar gene expression patterns and providing links to the scientific literature reporting genes with particular expressions.

We have been very encouraged by the scalability and reliability of the state-of-the-art Semantic Web tools and technologies. The biggest scalability problem we encountered initially was when trying to stream large SPARQL query results, a feature now well supported by the latest version of Jena TDB. Our SPARQLite implementation enabled us to develop fine-grained control of SPARQL query policies implemented for our SPARQL endpoints, thereby avoiding quality-of-service problems that might otherwise have arisen.

Acknowledgment

This work was supported by funding from the JISC to Dr David Shotton for the FlyWeb Project (<http://flyweb.info>). We greatly acknowledge the help provided by Andy Seaborne, Semantic Web Team, HP Labs, Bristol; and by Dr Helen White-Cooper, School of Biosciences, University of Cardiff.

References

1. Belleau, F., Nolin, M., Tourigny, N., Rigault, P., Morissette, J.: Bio2RDF: Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics* 41, 706–716 (2008)

¹¹ <http://openflydata.org>

2. Benson, E., Klyne, G., Gudmannsdottir, E., Shotton, D., White-Cooper, H.: The *Drosophila* testis gene expression database. In: The 47th Annual Drosophila Research Conference (2006)
3. Bizer, C., Cyganiak, R.: D2R Server—publishing relational databases on the semantic web (poster). In: Proceedings of the 5th International Semantic Web Conference, Athens, GA, USA (2006)
4. Chintapalli, V., Wang, J., Dow, J.: Using FlyAtlas to identify better *Drosophila melanogaster* models of human disease. *Nature Genetics* 39(6), 715 (2007)
5. Clark, K.G., Feigenbaum, L., Torres, E.: SPARQL protocol for RDF (2008), <http://www.w3.org/TR/rdf-sparql-protocol/>
6. Goble, C., Stevens, R.: State of the nation in data integration for bioinformatics. *Journal of Biomedical Informatics* 41(5), 687–693 (2008)
7. Jiang, J., White-Cooper, H.: Transcriptional activation in *Drosophila* spermatogenesis involves the mutually dependent function of *aly* and a novel meiotic arrest gene *cookie monster*. *Development* 130, 563–573 (2008)
8. Lyne, R., Smith, R., Rutherford, K., Wakeling, M., Varley, A., Guillier, F., Janssens, H., Ji, W., McLaren, P., North, P., et al.: FlyMine: an integrated database for *Drosophila* and *Anopheles* genomics. *Genome Biology* 8(7), R129 (2007)
9. Oinn, T., Greenwood, M., Addis, M., Alpdemir, M.N., Ferris, J., Glover, K., Goble, C., Goderis, A., Hull, D., Marvin, D., Li, P., Lord, P., Pocock, M.R., Senger, M., Stevens, R., Wipat, A., Wroe, C.: Taverna: Lessons in creating a workflow environment. *Concurrency and Computation: Practice and Experience* 18(10), 1067–1100 (2006)
10. Phuoc, D.L., Polleres, A., Tummarello, G., Morbidoni, C., Hauswirth, M.: Rapid Semantic Web mashup development through Semantic Web Pipes. In: Proceeding of the 18th International World Wide Web Conference, Madrid, Spain (2009) (to appear)
11. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF (2008), <http://www.w3.org/TR/rdf-sparql-query/>
12. Stein, L.: Integrating biological databases. *Nature Reviews Genetics* 4(5), 337–345 (2003)
13. Stein, L.: Towards a cyberinfrastructure for the biological sciences: progress, visions and challenges. *Nature Reviews Genetics* 9(9), 678–688 (2008)
14. Tomancak, P., Beaton, A., Weiszmam, R., Kwan, E., Shu, S., Lewis, S.E., Richards, S., Ashburner, M., Hartenstein, V., Celniker, S.E., Rubin, G.M.: Systematic determination of patterns of gene expression during *Drosophila* embryogenesis. *Genome Biology* 3(12), 81–88 (2002)
15. Tweedie, S., Ashburner, M., Falls, K., Leyland, P., McQuilton, P., Marygold, S., Millburn, G., Osumi-Sutherland, D., Schroeder, A., Seal, R., Zhang, H.: The FlyBase Consortium. FlyBase: enhancing *Drosophila* Gene Ontology annotations. *Nucleic Acids Research* 37, 5555–5559 (2009)
16. Van de Sompel, H., Nelson, M., Lagoze, C., Warner, S.: Resource harvesting within the OAI-PMH framework. *D-Lib Magazine* 10(12), 1082–9873 (2004)
17. Wilkinson, M., Links, M.: BioMOBY: An open source biological web services proposal. *Briefings in Bioinformatics* 3(4), 331–341 (2002)
18. Zhao, J., Klyne, G., Shotton, D.: Building a Semantic Web image repository for biological research images. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 154–169. Springer, Heidelberg (2008)

On the Reachability of Trustworthy Information from Integrated Exploratory Biological Queries

Eithon Cadag¹, Peter Tarczy-Hornoch¹, and Peter J. Myler^{1,2}

¹ University of Washington, Seattle, WA 98195

² Seattle Biomedical Research Institute, Seattle, WA 98109

{ecadag,pth}@u.washington.edu
peter.myler@sbri.org

Abstract. Levels of curation across biological databases are widely recognized as being highly variable, depending on provenance and type. In spite of ambiguous quality, searches against biological sources, such as those for sequence homology, remain a frontline strategy for biomedical scientists studying molecular data. In the following, we investigate the accessibility of well-curated data retrieved from explorative queries across multiple sources. We present the architecture and design of a lightweight data integration platform conducive to graph-theoretic analysis. Using data collected *via* this framework, we examine the reachability of evidence-supported annotations across triangulated sources in the face of uncertainty, using a simple random sampling model oriented around fault tolerance. We characterize the accessibility of high-quality data from uncertain queries and levels of redundancy across data sources and find that generally encountering non-experimentally verified annotations are nearly as likely as encountering experimentally verified annotations, with the exception of a group of proteins whose link structure is dominated by experimental evidence. Finally, we discuss the prospect of determining overall accessibility of relevant information based on metadata about a query and its results.

1 Introduction

With the abundance of biological information available in nearly hundreds-to-thousands of sources, many biologists rely on triangulated approaches to answer questions or explore data. For example, to investigate the link between an as-yet obscure gene and some pathology, a scientist may search for known protein-protein interaction data of homologous sequences, or look up pathway information on similar genes in other, closely-related species. At an even more basic level, gene annotation, one of the most vital steps in studying a new genome or maintaining a current one, often necessitates the use of multiple sources of information [1]. In a case such as that above, where there is a considerable amount of uncertainty regarding the quality or trustworthiness of data being retrieved or analyzed, it often falls to the individual scientists' experience to determine how confident they are with a conclusion. In light of the ever-increasing volumes of

biological data queued for review, the manual effort required to both integrate, analyze and especially curate information has become Herculean in both size and importance. Indeed, recent research suggests that this task is impossible, manually [2].

Calls for automated and semi-automated data integration methods and techniques in biology have not fallen on deaf ears and many systems have been developed to address the needs of biologists whose research demands the use of multiple, heterogeneous, biological information sources. Integrative resources have become important query, management and analysis tools for scientists, and automated approaches to integration have helped to alleviate manual efforts in data curation. Many data integration systems even serve as analytical platforms for functional characterization, among other things [3,4,5,6].

Most computationally-based annotation systems follow the general pattern of examining various sources of data or analyses to arrive at some conclusion to functional annotation that then must be curated manually. Unfortunately, the amount of data that requires human inspection and curation grows proportionally to the amount of data that is integrated and analyzed. This problem is particularly stark in protein function assignment where out-of-date, poorly-curated or computer-determined annotations populate data sources, which in turn transitively encourages incorrect annotations to new proteins [7]. Another problem then arises in this case – how best to gauge confidence in the correctness of functional assignments, given that the sources from which they are derived are of variable quality. Integrative methods that provide biological information as well as an estimate of how much their assertions can be trusted may become invaluable in prioritizing scientists’ attentions.

In the following work, we explore this problem in the context of functional annotation, a biological task that is, by nature, fraught with uncertainty. Using explorative queries of protein sequences against cross-linked data sources, we determine the likelihood that well-curated data is reached in an attempt to reproduce how an annotator, human or machine, may assign protein function. We describe the architecture of the lightweight, path-based federated data integration system used to collect our data, whose results generate a query graph of cross-referencing entities.

Next, we present a fault tolerance-based model of measuring the resiliency of a query with respect to well-curated data; we apply this to paths from a protein sequence query to experimentally-derived (per evidence codes) Gene Ontology¹ (GO) terms as a test case. We examine the properties of the results in terms of the likelihood of reaching high-quality data, and test the effect that unavailability of various sources may have on the overall results. We then examine characteristics of query graphs that imply a higher chance of encountering trustworthy data in the face of ambiguity.

It is important to highlight that the focus of this work is not the development of a method that identifies specific query results as better than others; rather, we attempt to ascertain the accessibility of experimentally-determined information

¹ <http://www.geneontology.org>

across biological data sources, which may then have implications on any analyses done on the data thereafter. Previous research in evaluating data integration systems and data of questionable value has primarily focused on identifying individual and specific high-quality information amongst many results. At a fundamental level, the premise of this research is to model the quality of query results in general, as well as estimating the difficulty in differentiating between high- and low-quality data.

2 Related Work

There has been considerable work done in the past towards developing and optimizing data integration systems that exploit the graph structure of interlinked queries to mine data or analyze paths, although to our knowledge there are few, if any, that examine the graph in its entirety for the purposes of evaluating the overall quality of a query and the associated general reachability of high-quality data.

Among prior research that does deal with mining data from integrated biological sources, the work by Cohen-Boulakia, *et al.* with BioGuide, Lacroix *et al.* with BioNavigator and our previous work with BioMediator, stand out as the most comparable in terms of the data integration approach taken [8,9,10,11]. These systems rely on path-based querying over a graphical representation of the results, accomodate a browsing mode for exploratory queries and use explicit mappings of predefined entities to multiple sources for increased flexibility, all of which we use in the implementation described later.

Independent of the architecture, work by Lacroix, *et al.* leveraged graph statistics such as path length and cardinality to devise efficient query search plans and paths [12]. This research is related to our own, in that general characteristics of the graph are studied, although our work differs as we try to characterize overall result quality and reachability, as opposed to overall expected result volume or path rank. Relatedley, Detwiler *et al.*, developed the BioRank system, which used data provenance to rank integrated data by relevance [13]; and Louie *et al.* use a random sampling model to determine individual node belief probabilities, although they found that the prime utility of the beliefs were to rank results within the same query, and not a measure of the overall absolute quality of the data retrieved [14]. The problem we attempt to address, the question of overall accessibility of curated data in the face of noisy biological databases, is complementary to this.

In regards to research on the reliability and tolerance of networks to failure, there has been extensive work on both theoretical and naturally-occurring graphs. Albert, *et al.*, for instance, demonstrate the robustness of scale-free networks against random failure when compared to binomial networks, and Amaral, *et al.* showed that neural connectivity follows this pattern [15,16]; Cohen, *et al.* studied the resilience of the Internet to random breakdown using percolation theory by incrementally increasing the rate of failure, an approach similar to the one we take here [17]; and Searls found that database schemata and software

dependency at least partially display robustness against failure [18]. Our work adopts this paradigm of connectivity as applied to measuring access to curated data amongst integrated biological sources.

Finally, a similar area of research to our own is the field of scientific workflow modeling and representation. Workflow systems such as Taverna [19] allow scientists to visually compose pipelines of actions upon input data to produce result outputs. From a structural standpoint, our integration system represents data in a similar manner, and its operation can be described as a very basic workflow execution system where actionable tasks are limited to only protein sequence queries and joins. However, whereas many workflow systems have been adapted and designed to track provenance [20], for our system and experiments described below, we omit most considerations for evidence that may be available in the graphs, reachability excluded.

3 Methods

3.1 Data Integration for Exploring Graphical Models

We developed the *MIQAPS* framework (for “**M**ultisource **I**ntegrated **Q**ueries **A**gainst **P**rotein **S**equences”, pronounced *mai-kaps*) as a query, integration and retrieval engine for exploratory queries. Using *MIQAPS*, it is possible to query using a protein sequence and retrieve similarity-based results across different heterogeneous resources, a practice that is amongst the most basic of needs in modern molecular biology laboratories.

Relying on a general-purpose data integration engine², *MIQAPS* is similar to *BioMediator* in architecture, with some notable operational differences, the largest of which are that it was designed to be extremely lightweight and low-overhead (core codebase is ≈ 2900 lines of Python), and supports caching within a relational database, *via* PostgreSQL³. Its scaled, extensible architecture allows for easy coupling with other systems or components, and for our work we joined *MIQAPS* with a graph analysis engine⁴ so that results from queries in *MIQAPS* are automatically mapped onto a manipulable graph for examination. This allowed us to conduct reachability experiments on a query graph of results.

A query is executed in the following manner: a client seeds a sequence query to the *browser* in the form of an entity-attribute-value tuple, *e.g.*, (*ProteinSequenceQuery*, *Sequence*, “*MRWAQ...*”), valid types of which are enumerated within the *schema*. The *schema* is frame-based, and thus hierarchical, and attributes of a parent are automatically inherited by a child entity (*e.g.* *ProteinSequenceQuery ISA Query*). The *browser* communicates with the *data directory*, which performs a look-up within the *source catalogue* of the data sources to determine, given the tuple, which may be queried. The *data directory* manages the individual interfaces to each data source (denoted by *DB x* in Figure 1), which are modularized and separate from the core engine and whose translation from source

² PyDI, <http://pydi.sourceforge.net>

³ <http://www.postgresql.org>

⁴ NetworkX, <http://networkx.lanl.gov>

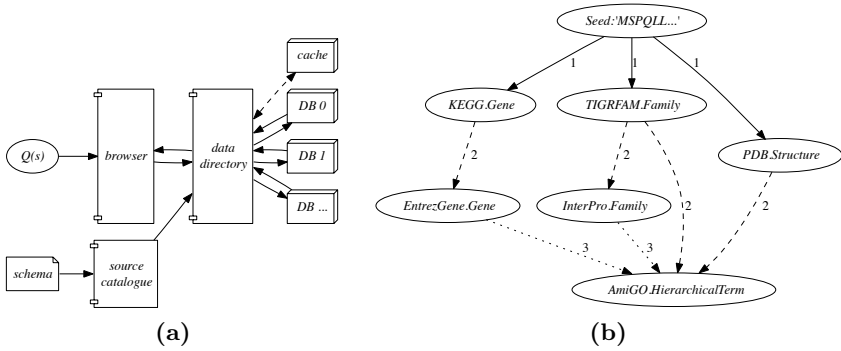


Fig. 1. Architecture of the MIQAPS system (1a); arrows denote information flow between the various components (dotted indicates a non-required interaction). 1b is an example of the query-expansion mechanism; a query is seeded with some sequence (top), and expanded to the first set of sources (solid arrows); results from these sources are then expanded *via* references to the next set (dashed arrows), and so on. This may continue until further expansion is not possible.

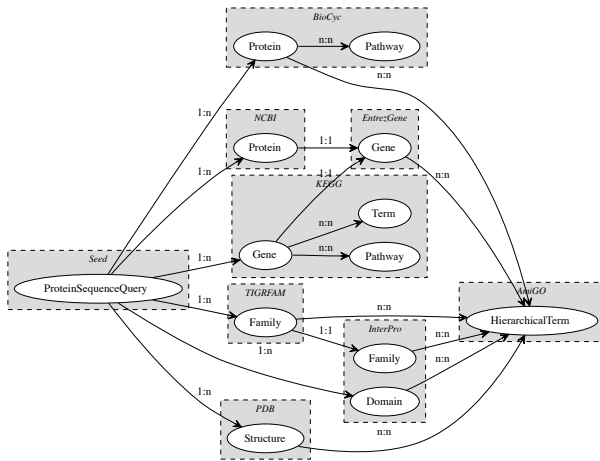


Fig. 2. The above shows the data sources (in grey, dotted boxes), associated classes (ellipses), cross-links (arrows) and cardinalities (arrow labels) for the integrated data retrieval system. Queries begin on the left, with the *Seed* source, and are expanded rightward.

to MIQAPS is declared in the schema in the form of mappings. Data that is returned may themselves be queryable, and thus continued querying of results amounts to sequential joins across numerous databases. Once translated by the data directory into materialized schema entities, attributes and relations, results are sent back to the browser in the form of a *query graph* $G = (V, E)$, where nodes V represent individual data records mapped to entities in the schema, and $E \subset \{V \times V\}$ the directed relations between those entities. Note that given this

structure, there is a path from the seeded query to all other nodes; that is, all nodes are *reachable* from the initiating query.

Outlined in Figure 2 are the linkage relationships from the schema we used to represent our graphs with respect to GO terms, represented in the figure as the right-most source-entity combination, *AmiGO*⁵ / *HierarchicalTerm*. Note that query graphs generated by this schema are acyclic. The behavior of the query graph retaining paths to GO terms despite a range of node and source unavailability or removal is our primary interest, and so all sources lead to *AmiGO*. There are also a number of sources which link to other non-*AmiGO* sources that we include to model the effects of redundancy that occur naturally in biological databases that cross-reference one another.

3.2 A Fault Tolerance Model over Arbitrary Query Graphs

For a query graph G seeded with a protein sequence and generated using the data integration methods described above, we are now interested in how well connectivity holds from a seed node $s \in V$ to any GO term $t \in V$. Below, we describe the method used to estimate the reachability of t from s , or $s \rightsquigarrow t$ ⁶ under arbitrary conditions using random sampling.

Let $R \in V$ represent the set of nodes for which we want to test reachability from s (R may be the set of experimentally-validated data, for instance), and l a specified failure rate – the probability that any random node $n \in V \setminus (R \cup \{s\})$ ⁷ is unavailable, whereupon n and edges entering and leaving n are removed from G , recursively. This model emulates instances where a data record did not return as expected, due to source downtime, retrieval processing problems or errors of omission on the part of the scientist. We denote the event of removing nodes as inducing node *failure* F . Then we define the probability that any node $t \in R$ retains a path from s after F as:

$$p(G, R, l) \equiv P(s \rightsquigarrow t | F_l). \quad (1)$$

The algorithm in Figure 3 executes the recursive node removal process using random and repeated simulations, thus generating an approximation of the probability expressed in (1).

Extending (1) to all possible failure rates and taking the area under the resulting curve gives the *fault tolerance* of a graph G given arbitrary nodes R with respect to l . Formally, we express the fault tolerance, τ , of a query graph by:

$$\tau(G, R) = \int_{-\infty}^{\infty} p(G, R, l) dl. \quad (2)$$

⁵ <http://amigo.geneontology.org>

⁶ Where $s \rightsquigarrow t$ denotes that a transitive path exists from s to t .

⁷ Where $A \setminus B \equiv \{x \in A | x \notin B\}$.

Input: $G = (V, E)$ with initial query $s \in V$, targets $R \subset V$, simulations $i \in \mathbb{Z}^+$, failure rate $l = (0, 1)$

Result: tolerance value $r = (0, 1)$ for G, R w.r.t. failure rate l

```

1  $m \leftarrow []$ 
2 for iteration  $j, i$  times do
3    $H = (V', E') \leftarrow G, S \leftarrow R$ 
   // Sample random nodes for removal (excluding the provided targets and seed)
4   for  $|V'| * l$  times do  $V' \leftarrow V' \setminus \text{sample}(V' \setminus (R \cup \{s\}))$ 
5   for  $t \in S$  do
6     if  $\neg(s \rightsquigarrow t)$  in  $H$  then  $S \leftarrow S \setminus \{t\}$ 
7   end
8    $m_j \leftarrow \frac{|S|}{|R|}$ 
9 end
10 return  $(\sum_k m_k)/i$ 

```

Fig. 3. Sampling procedure used to approximate $p(G, R, l)$ with respect to target nodes R , in query graph G , at a failure rate l ; applying this with $l = (0.0, \dots, 1.0)$ yields $\tau \approx \frac{1}{|l|} \cdot \sum_{k \in l} p(G, R, k)$.

Note that as $p(G, R, l)$ defines a probability function, $\tau(G, R)$ is bounded from below by 0 and above by 1. In practical terms, one can imagine τ as the probability that an annotator, following links from biological data source to data source and who may choose to abandon a path or select one path over another, will reach any term within a pre-specified set of GO terms R .

We denote the tolerance curve of our query graphs hereafter as τ_{exp} , with the above R the set of experimentally-determined GO terms. It is also desirable to generate the *random tolerance curve* $\tau_{rdm}(G, N)$, where $N \subset V \setminus \{s\}$ and $|N| = |R|$, with the latter property used when in comparison with τ_{exp} . Calculation of τ_{rdm} would proceed in a similar way as τ_{exp} with the exception that nodes in N may be chosen randomly and at each iteration. Furthermore, we allow the possibility that nodes in R and any given N may overlap. The random tolerance curve provides a baseline level of average reachability for G with which to compare against the experimental tolerance curve, and N may be set to a specific set of nodes in a particular domain, *e.g.*, such as when comparing well-curated GO terms, R , against poorly curated ones, placed in N .

In the context of integrated biological data sources, this allows us to measure how well-curated data is connected to an original query *via* any number of paths and links of questionable relevancy. For example, data that is known *a priori* to be of higher-quality that retains reachability from s in the face of random failure, while at the same time lower-quality data loses reachability, suggests a query in which one can assign a reasonable level of confidence that sound results can be easily separated from unsupported results. On the other hand, graphs where lower-quality data retains reachability from the query better than high-quality data imply that separating the “wheat from the chaff” may involve a great deal of effort. As we use a method based on simple random sampling, we can easily adapt to examining subparts of a query graph, such as inducing failure only upon nodes that originate from a particular source; this enables us to explore the effects of data reachability in the face of specific source unavailability (a situation not uncommon for federated data integration systems) or omission.

3.3 Estimating Tolerance of Biological Queries for High-Quality Data

We apply the model formalized in Section 3.2 to query graphs generated by MIQAPS. For this purpose, we randomly selected a small number of *UniProt* accessions from the *Gene Ontology Annotation Database*⁸ (*GOA*) *UniProt* which contained GO annotations made on the basis of experimental evidence, as outlined *via* GO evidence codes⁹ in *GOA* (e.g., ‘EXP’, ‘IGI’). GO codes annotated in this fashion were considered to be experimentally verified, while all others (e.g., ‘ISS’, ‘TAS’) were nominally treated as not experimentally validated (non-verified). Thus, within an actual query and for evaluation purposes a GO term may be labeled as experimental, as determined by *GOA*, even if the associated evidence code in the query graph itself is not experimental.

We identified 104 proteins whose annotations included experimental codes, and which returned GO terms when queried using MIQAPS. Per Figure 2, we relied on seven different data sources to reach GO terms from our initial sequence query. The sources selected are well-known biological repositories searchable *via* protein sequence, and provide coverage over several different biological domains:

- Genes, Proteins: *EntrezGene*¹⁰, *EntrezProtein* (via *BLAST*)¹¹, *BioCyc*¹², *KEGG*¹³
- Pathways: *BioCyc*, *KEGG*
- Domains, Families: *TIGRFAM*¹⁴, *InterPro*¹⁵
- Structural data: *PDB*¹⁶

Caveats to the approach described above include the possible confounder that annotations linked from one of the sources MIQAPS covers contains GO terms transitively assigned from *GOA UniProt*. We attempted to mitigate this by naturally excluding *UniProt* from our list of sources above; unfortunately, this problem is difficult to avoid when dealing with protein annotations across databases that are often referenced to curate other databases. Additionally, though we treated *GOA*-determined experimental GO terms as our evaluation standard and the sole measure of high-quality data, other GO terms may be equally valid annotations, though not experimentally verified according to *GOA*.

Because estimation of (II) is based on a simple random sampling approach that requires a large number of simulations, we tested for the optimum choice of simulations that was not overly time-expensive but still converged on a value for τ . Using 11 randomly-selected proteins from our set of 104, we found that

⁸ <http://www.ebi.ac.uk/GOA>

⁹ GO evidence codes at <http://www.geneontology.org/GO.evidence.shtml>

¹⁰ <http://www.ncbi.nlm.nih.gov/sites/entrez?db=gene>

¹¹ <http://blast.ncbi.nlm.nih.gov/Blast.cgi>

¹² <http://biocyc.org>

¹³ <http://www.genome.jp/kegg>

¹⁴ <http://www.jcvi.org/cms/research/projects/tigrfams>

¹⁵ <http://www.ebi.ac.uk/interpro>

¹⁶ <http://www.rcsb.org/pdb>

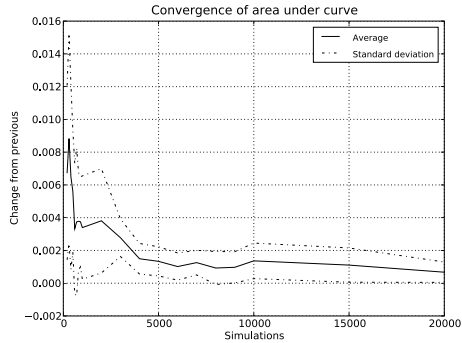


Fig. 4. Based on 11 randomly selected queries, convergence of the fault tolerance measure τ started at approximately 10 000 simulations

on average convergence began at approximately 10 000 simulations (see Figure 4); this was thus the value we used for i in running the algorithm in Figure 3.

In addition to comparing the behavior of the tolerance curves and fault tolerance values τ_{exp} and τ_{rdm} in the form of $\tau_{exp-rdm} = \tau_{exp} - \tau_{rdm}$, we also examined general graph statistics, such as the number of nodes and edges, the radius, the number of nodes originating from each source and the number of recognized *a priori* experimental GO terms, among others. We further obtained the probability that a GO term t is experimentally verified, given that there exists a node from a non-*AmiGO* source I which has a path to t , *i.e.*, $P(t \in R|I \rightsquigarrow t)$; this was readily-calculable from the query graphs themselves.

4 Results

Using the methods described in sections 3.2 and 3.3, we generated the τ_{exp} and τ_{rdm} curves for each of our 104 sample proteins. Overall, we found that the τ value was greater for the experimental GO terms than random GO terms in 60.5% of the cases, which implies that curated and verified annotation data will tend to be referenced more often in data sources than non-verified annotations. On closer detail, however, there was notable variety in how well an individual query was connected to a GO term through the data sources; values for τ_{exp} ranged from 0.348 to as high as 0.882, and τ_{rdm} 0.348 to 0.790. Figure 5 shows four different tolerance curves from the test set, representative of the curves generated overall; τ_{exp} ranges from extremely tolerant against node failure to slightly-worse than the proportional failure rate.

The variation is made clearer when the average curves of τ_{exp} and τ_{rdm} are compared across all 104 proteins (see Figure 6). The average difference between the two curves is only marginal, although τ_{exp} is significantly more varied towards the upper end. A possible explanation for this is that there are a minority of proteins where the curation toward experimentally verified results has been more thoroughly propagated throughout many of the data sources, and so there are

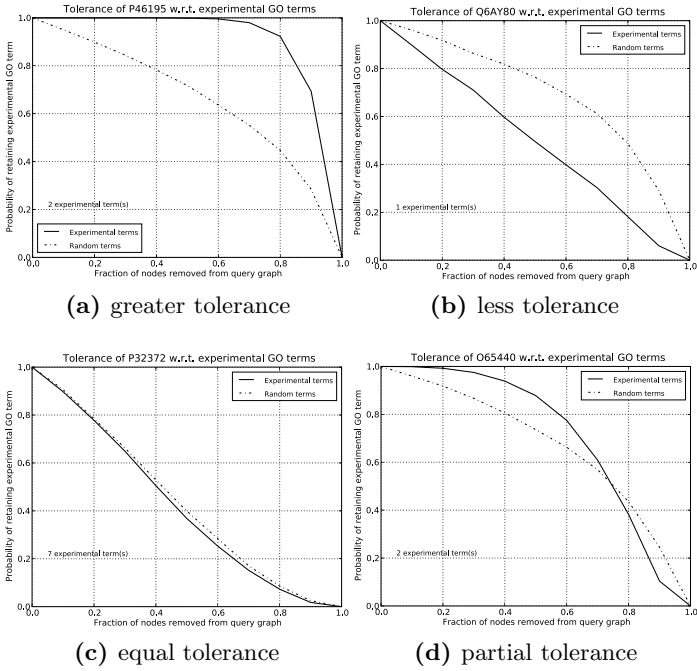


Fig. 5. Selected archetypal tolerance curves (τ_{exp}) for experimentally-validated Gene Ontology terms compared to non-experimental (τ_{rdm})

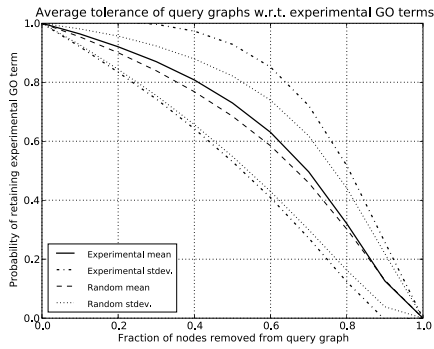


Fig. 6. Average tolerances and standard deviations between experimental GO terms and random GO terms

numerous paths to the term from the query. In this group, even with as high as a 50% node failure rate across the query graph, a number of experimental GO terms remain reasonably reachable. For other proteins, however, it is difficult to discern the experimentally verified GO terms from others based on data source cross-linking alone.

Table 1. Top 10 and bottom 10 in $\tau_{exp-rdm}$; proteins where the experimental GO terms are likely to stand out are from mammalian proteomes, or have been reviewed

<i>Accession</i>	$\tau_{exp-rdm}$	<i>Species</i>	<i>Reviewed?</i>
Q9JIL4	0.249	<i>M. musculus</i>	×
Q49IK6	0.242	<i>D. rerio</i>	
P46195	0.225	<i>B. taurus</i>	×
O88379	0.199	<i>M. musculus</i>	×
Q61176	0.184	<i>M. musculus</i>	×
O55081	0.175	<i>R. norvegicus</i>	×
P55209	0.173	<i>H. sapiens</i>	×
Q14AC4	0.165	<i>M. musculus</i>	
Q10176	0.154	<i>S. pombe</i>	×
A2A602	0.148	<i>M. musculus</i>	
Q2KT22	-0.194	<i>D. rerio</i>	
Q19328	-0.192	<i>C. elegans</i>	
Q8AY90	-0.183	<i>D. rerio</i>	
Q6AY80	-0.178	<i>R. norvegicus</i>	×
A4FVK4	-0.175	<i>D. rerio</i>	
P46974	-0.142	<i>S. cerevisiae</i>	×
P91621	-0.127	<i>D. melanogaster</i>	×
Q24133	-0.111	<i>D. melanogaster</i>	×
B3DFT2	-0.104	<i>D. rerio</i>	
Q7JLC3	-0.093	<i>C. elegans</i>	

To explore this further, we examined the graph statistics of the top 10 and bottom 10 query graphs, as sorted by $\tau_{exp-rdm}$ in Table 1. Interestingly, the query graphs where $\tau_{exp-rdm}$ is greatest were *smaller* than those where $\tau_{exp-rdm}$ were least, in terms of absolute number of edges and in ratio to the number of nodes (2.80 edges/node versus 2.34); the queries where the experimental results were most accessible were less connected than those where the experimental terms were indistinguishable from random terms, path-wise.

In terms of biological relevance, we found that the queries with a large $\tau_{exp-rdm}$ tended to be entries in *UniProt* that were reviewed (70%), half of which originating from the mouse proteome. Conversely, queries where $\tau_{exp-rdm}$ were most negative were more likely not to be reviewed (40%), and come from a more phylogenetically diverse list of organisms. As the mouse model is commonly used in biomedical research, the greater likelihood of arriving at an experimentally-based conclusion in the former is not surprising. Indeed, high $\tau_{exp-rdm}$ queries contained more experimental GO terms in their query results than low $\tau_{exp-rdm}$ queries, although there was no strong correlation between $\tau_{exp-rdm}$ and the number of experimental GO terms among the 104 queries overall.

In addition to having fewer nodes and edges, the top 10 experimental queries also had a smaller number of average incoming edges to *AmiGO* terms, specifically 2-3.65 versus 3-5. At first glance this may seem counterintuitive, but a possible explanation may be that these protein and those similar to it occupy

Table 2. Average query graph tolerances for experimental Gene Ontology terms with source-targeted node failure, and likelihoods that a term is experimentally-derived, given that a source has a path to it

Data source (I)	Exp. GO terms (τ_{exp})	Rand. GO terms (τ_{rdm})	$P(t \in R I \rightsquigarrow t)$
<i>BioCyc</i>	1.000	0.966	0.052
<i>EntrezGene</i>	0.710	0.729	0.164
<i>InterPro</i>	0.999	0.994	0.101
<i>KEGG</i>	1.000	0.994	0.167
<i>PDB</i>	0.998	0.969	0.053
<i>TIGRFAM</i>	0.993	0.994	0.070

a desirable position for highlighting experimental evidence – a balanced trade-off between redundancy and coverage where high-quality data is dominant and likely to be encountered, and is not drowned out by the presence of other information. Results thus far have been in the case of *random* failure – that is, any node from any source having a fractional l chance being lost, much like how an annotator may miss or choose to ignore pursuing links while characterizing a protein. We were also interested in omissions of entire sources, as well, and the effect that may have on reaching supportive annotation data. To this end, we “knocked out” each of the sources that link to GO terms individually along the same linear failure rate $l = (0.0, \dots, 1.0)$ and reviewed the effects incremental loss of individual sources had on tolerance.

Table 2 shows the results of individual source unavailability; τ is significantly higher in these instances as we only remove a single source at a time, and the rest of the query graph remains untouched. It immediately stands out that in the case of all but one of the sources (*EntrezGene*), the query graphs generally retain connectivity to GO terms, both random and experimental; loss of *EntrezGene* reduces τ significantly. However, *EntrezGene* nodes constitute on average 43% of the nodes in the query graphs that link to GO terms, yet have an impact of only 27-29% when removed, suggestive of considerable redundancy and overlap amongst the sources collectively.

Examining the probabilities that a source will lead to an experimental GO term, given that it leads to any GO term, provides additional information in regards to the overlapping coverage between the databases. For example, $P(t \in R | I \rightsquigarrow t)$ is higher in *KEGG* than in *EntrezGene*, yet incrementally removing *KEGG* decreases the likelihood of encountering random, non-experimental terms, whereas the reverse is true for *EntrezGene*. A hypothesis to explain this occurrence may be that as a source, *KEGG* is likelier to reference GO terms to which other sources already link, and thus is useful for redundant coverage, but is less effective at uncovering novel, experimental data. *EntrezGene*, on the other hand, appears to be a rich source for experimental GO terms, and although it had relatively few links to GO terms by comparison to make a significant impact, *TIGRFAM* possibly shares similar attributes.

Finally, we compared the overall graph structure of query graphs with well-known graph structures of similar composition – namely, binomial (Erdős-Rényi) graphs, and power law graphs – with the intent to measure how graphs built from cross-references of disparate sources compare in resiliency to other graphical models. The representations of the canonical graphs were generated such that the node and edge counts, likelihood of edges between nodes and size $|N|$ were similar to our query graph averages. Seed nodes in these graphs were simulated by randomly selecting a node in the center of the graph, and random nodes on the periphery were chosen as our ersatz “GO terms.” (see Figure 7).

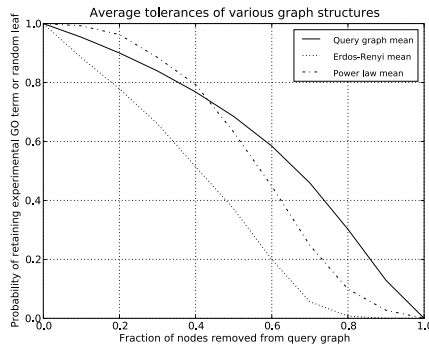


Fig. 7. Comparison of the random tolerance curve from query graphs, to the tolerance of similarly-composed power law and Erdős-Rényi graphs. Above, $\tau_{rdm} = 0.601$, $\tau_{pow} = 0.553$ and $\tau_{erg} = 0.406$.

Not surprisingly, binomial graphs lost connectivity to the peripheral nodes rapidly, as edges were formed independently and the formation of strongly-connected clusters or hubs was unlikely (incidentally, a number of individual query graphs appeared to follow this model as well). At the opposite end of the spectrum, the power law graph maintains resiliency over the average query graph for $l \leq 0.45$ before it is overtaken. This is an interesting finding, as it suggests that some query graphs appear to have properties of resiliency equal to or greater than that of power law or scale-free networks. We are, however, reticent to generalize the resiliency of query graphs beyond the narrow seed-to-term focus we provide here as the sources and references we use are specifically geared towards forming paths to a targeted set of nodes.

5 Discussion

In this work, we developed a simple model to measure with what relative likelihood an ambiguous protein query can reach relevant experimental data for annotation purposes, and developed a system to test the reachability of GO terms across multiple data sources from the query. Our working hypothesis was that experimentally verified, high-quality information would be reachable from

a query through many more paths than non-verified information in the space of integrated biological data sources.

What we found, however, was that an annotating scientist relying purely on sequence-based searches is almost as likely to find redundant evidence for non-experimentally-derived terms for protein function as experimental. This is not to say that non-experimentally-derived information is categorically less relevant when compared to one that is experimentally-derived; rather, that there are non-trivial challenges to unearthing strongly-corroborated data amongst the multitudes of biological information available, *even* when queries are triangulated from multiple sources. We speculate that reasons for these unanticipated findings, in addition to those we outline specifically in our results, are generally related to the inherent nature of the data, sources and domain we tested. When curation level varies for a single protein, so too may the curation of others whose annotations are influenced by that protein.

Our exploratory study used well-curated *GOA* annotations as a gold standard. However, many *ad hoc* computational approaches to protein annotation do not go beyond basic searches of one or two data sources, and there is a danger that many trustworthy, verifiable annotations are ignored or never considered in face of the abundance of data that populate many biological resources. Part of our results indicated that when compared to power law networks, schemata and queries designed for specific retrieval of annotation terms can exhibit surprising robustness, as it pertains to the narrow focus of protein annotation terms. Our findings suggest, however, that this robustness may be a double-edged sword, as validated information may be buried beneath volumes of other data.

In order to accomplish more thorough queries of biological data manual searches are insufficient, and automated approaches sorely needed. However, these automated methods are liable to generate massive amounts of data that must then be filtered and prioritized. While our results echo the conventional wisdom that biological databases are rife with noise, we are encouraged to find that there may be ways of estimating the amount of spurious information, and thus provide an opportunity to prioritize tasks to those which likely need manual review the most or least. Our findings showed that the queries most likely to point to experimental, verified data more than non-verified shared some similar graph characteristics, including graph sizes and term in-degrees within a certain range. Graph metrics such as these may be amenable to principled approaches to assigning a measure of confidence to the quality of data retrieved (*e.g.*, τ , in our case); well-trained and noise-insensitive statistical methods may be able to predict the overall trustworthiness of a result set based on the characteristics and structure of the query graph. Furthermore, as it is generally known and reaffirmed in our findings, not all genomes and data sources contain information that are necessarily experimentally verified at comparable rates, and biological data integration systems that are sensitive to this may be able to greatly minimize the amount of noise encountered when executing uncertain queries.

The application of these methods, as well as tests for how well fault tolerance-based measures for predicting result uncertainty generalize beyond protein

annotation data, are future work. Biomedicine is replete with examples of the need for methods to help manage nebulous, overwhelming information from queries of myriad sources. There is an opportunity even before the analysis of any individual result of a query to estimate the level of confidence for what is returned. This knowledge may better help inform consumers of the data how the information may best be used, analyzed and prioritized; an approach for measuring access to trustworthy data and a framework for testing it is a first step.

Acknowledgements. The authors would like to thank the anonymous reviewers for their thoughtful and constructive feedback. This research is supported by the National Library of Medicine training grant T15LM07442, and National Science Foundation grant IIS-0513877.

References

1. Garrels, J.: Yeast genomic databases and the challenge of the post-genomic era. *Fun. Integr. Geno.* 2, 212–237 (2002)
2. Baumgartner, W., Cohen, K., Fox, L., Acquaah-Mensah, G., Hunter, L.: Manual curation is not sufficient for annotation of genomic databases. In: *ISMB/ECCB (suppl. of Bioinform.)*, pp. 41–48 (2007)
3. Kasukawa, T., Furuno, M., et al.: Development and Evaluation of an Automated Annotation Pipeline and cDNA Annotation System. *Gen. Res.* 13 (2003)
4. Potter, S., Clarke, L., et al.: The ENSEMBL Analysis Pipeline. *Gen. Res.* 14 (2004)
5. Shah, S., Huang, Y., Xu, T., Yuen, M., Ling, J., Ouellette, B.: Atlas - a data warehouse for integrative bioinformatics. *BMC Bioinform.* 6, 34 (2005)
6. Strothard, P., Wishart, D.: Automated bacterial genome analysis and annotation. *Curr. Opin. Microbiol.* 9(5), 505–510 (2006)
7. Bork, P.: Powers and pitfalls in sequence analysis: the 70% hurdle. *Gen. Res.* 10(4), 398–400 (2000)
8. Mork, P., Halevy, A., Tarczy-Hornoch, P.: A Model for Data Integration Systems of Biomedical Data Applied to Online Genetic Databases. In: *AMIA 2001*, pp. 473–477 (2001)
9. Cohen-Boulakia, S., Davidson, S., Froideveaux, C., Lacroix, Z., Vidal, M.: Path-based systems to guide scientists in the maze of biological data sources. *J. Bioinform. Comput. Biol.* 4(5), 1069–1095 (2006)
10. Lacroix, Z., Parekh, K., Vidal, M., Cardenas, M., Marquez, N.: BioNavigation: Selecting Optimum Paths Through Biological Resources to Evaluate Ontological Navigational Queries. In: Ludäscher, B., Raschid, L. (eds.) *DILS 2005. LNCS (LNBI)*, vol. 3615, pp. 275–283. Springer, Heidelberg (2005)
11. Shaker, R., Mork, P., Brockenbrough, J., Donelson, L., Tarczy-Hornoch, P.: The BioMediator System as a Tool for Integrating Biologic Databases on the Web. In: *Worksh. IIW, VLDB* (2004)
12. Lacroix, Z., Raschid, L., Vidal, M.: Efficient Techniques to Explore and Rank Paths in Life Science Data Sources. In: Rahm, E. (ed.) *DILS 2004. LNCS (LNBI)*, vol. 2994, pp. 187–202. Springer, Heidelberg (2004)
13. Detwiler, L., Gatterbauer, W., Louie, B., Suci, D., Tarczy-Hornoch, P.: Intergrating and Ranking Uncertain Scientific Data. In: *ICDE 2009* (2009) (to appear)

14. Louie, B., Detwiler, L., Dalvi, N., Shaker, R., Tarczy-Hornoch, P., Suciu, D.: Incorporating uncertainty metrics into a general-purpose data integration system. In: SSDBM, p. 19 (2007)
15. Albert, R., Jeong, H., Barabasi, A.: Error and attack tolerance of complex networks. *Nat.* 406, 378–382 (2000)
16. Amaral, L., Scala, A., Barthélémy, S.H.: Classes of small-world networks. *Proc. Nat. Acad. Sci.* 97(21), 11149–11152 (2000)
17. Cohen, R., Erez, K., ben-Avraham, D., Havlin, S.: Resilience of the Internet to Random Breakdowns. *Phys. Rev. Lett.* 85, 4626–4628 (2000)
18. Searls, D.: Data integration – connecting the dots. *Nat. Biotech.* 21, 844–845 (2003)
19. Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M., Li, P., Oinn, T.: Taverna: a tool for building and running workflows of services. *Nuc. Ac. Res.* 34(Web Server issue), 729–732 (2006)
20. Davidson, S., Freire, J.: Provenance and Scientific Workflows: Challenges and Opportunities. In: SIGMOD 2008, pp. 1345–1350 (2008)

Estimating the Quality of Ontology-Based Annotations by Considering Evolutionary Changes

Anika Gross¹, Michael Hartung¹, Toralf Kirsten^{1,2}, and Erhard Rahm^{1,3}

¹ Interdisciplinary Centre for Bioinformatics, University of Leipzig

² Institute for Medical Informatics, Statistics and Epidemiology,
University of Leipzig

³ Department of Computer Science, University of Leipzig
{gross, hartung, tkirsten}@izbi.uni-leipzig.de,
rahm@informatik.uni-leipzig.de

Abstract. Ontology-based annotations associate objects, such as genes and proteins, with well-defined ontology concepts to semantically and uniformly describe object properties. Such annotation mappings are utilized in different applications and analysis studies whose results strongly depend on the quality of the used annotations. To study the quality of annotations we propose a generic evaluation approach considering the annotation generation methods (provenance) as well as the evolution of ontologies, object sources, and annotations. Thus, it facilitates the identification of reliable annotations, e.g., for use in analysis applications. We evaluate our approach for functional protein annotations in Ensembl and Swiss-Prot using the Gene Ontology.

Keywords: annotation, evolution, quality.

1 Introduction

Ontologies and their application have become increasingly important especially in the life sciences. Typically, they are used to semantically describe or annotate properties of real world objects, such as genes and proteins. The associations between object descriptions and the elements (concepts) of an ontology form a so-called *annotation mapping*. For instance, the protein objects of Ensembl [11] and Swiss-Prot [3] are associated with concepts of the popular Gene Ontology [9] to describe the molecular functions and biological processes in which the proteins are involved. Annotation mappings are utilized in different analysis scenarios and applications. These include functional profiling of large datasets such as gene expression microarrays (e.g., [1,4]), network reconstruction and retrieval [7], or instance-based ontology matching [13].

Computed results of these applications significantly depend on which annotations are used and hence rely on a good quality of the annotations, e.g., with respect to their correctness and completeness. A particularly important quality aspect is the stability of annotations since major changes in the annotation mappings may substantially influence or even invalidate earlier findings. This is potentially a major issue since annotation mappings change frequently, e.g., due to changes (additions, deletions,

Instance ID	Concept ID	V ₄₈	V ₄₉	V ₅₀	V ₅₁	V ₅₂
ENSP00000344151	GO:0015808 (L-alanine transport)	IDA	IDA	IDA	IDA	IDA
ENSP00000230480	GO:0005615 (extracellular space)	TAS	TAS	IDA	TAS	IEA
ENSP00000352999	GO:0006915 (apoptosis)	IDA	-	-	-	IDA

Fig. 1. Evolution of functional protein annotations in Ensembl versions (V₄₈-V₅₂ = Dec.2007-Dec.2008)

modifications) in the underlying ontologies [10], objects and annotation associations. Furthermore, annotation quality is influenced by the method that has been used to create the annotation because it likely affects how biologically founded or reliable an annotation is. The relevance of the creation method is underlined by the increasing use of predefined *evidence codes* (EC) to classify functional annotations based on the Gene Ontology [8]. These evidence codes allow a distinction of whether annotations are experimentally founded, are based on author or curator statements or generated by automatic algorithms, e.g., data mining techniques or homology mappings. The evidence codes represent *provenance* information (sometimes also called lineage¹ [2,5]) that can be utilized by analysis applications to focus on specific annotation sets, e.g., manually curated or automatically generated annotations.

For illustration, Figure 1 shows the evolution of selected functional protein annotations in five succeeding Ensembl versions (V₄₈-V₅₂). The first annotation (ENSP00000344151, GO:0015808) was continuously available with unchanged evidence code (IDA, *inferred from direct assay*) indicating a stable annotation. Conversely, the evidence code of the second annotation for protein ENSP00000230480 has been changed from *traceable author statement* (TAS) over IDA to *inferred from electronic annotation* (IEA). Such a frequent revision of the provenance information indicates reduced reliability of the annotation. Furthermore, the last annotation (Figure 1, line 3) was temporarily absent also indicating a reduced stability.

So far, the quality of annotation mappings w.r.t. their stability and provenance information is largely unexplored despite their potential importance for many analysis applications. We therefore present and evaluate a general approach to analyze annotation mappings by taking their evolution and evidence information into account. To that end we first propose an evolution model for annotation mappings including change operators and quality measures (Section 2). The model captures ontology, instance and quality changes w.r.t. annotation changes. Based on the evolution model, we propose evolution-based quality measures to identify reliable annotations (Section 3). Finally, we evaluate our evolution model by comparatively analyzing the annotation evolution in two large life science annotation sources, namely Ensembl and Swiss-Prot (Section 4). In particular, we study typical annotation changes and classify current annotations by applying the proposed assessment method. Section 5 discusses related work before we conclude.

The analysis results and the proposed assessment method for annotations are expected to be valuable for users and applications of life science annotations. In particular, algorithms may utilize information of annotation history and annotation quality to derive more robust / reliable results.

¹ We further use the term provenance to determine the original source of data.

2 Annotation Models

The stability of annotation mappings is affected by the changes in the involved instance (object) sources, ontologies and object-ontology associations. In the following we first introduce our model of annotation mappings including models for instance sources, ontologies and annotation quality. We will assume that annotations (object-ontology associations) include several quality indicators whose values may be taken from predefined quality taxonomies. In Section 2.2 we will introduce our evolution model including change operators for instances, ontologies and annotations. Furthermore, measures are proposed in order to quantify the evolution of annotations.

2.1 Annotation Mapping and Quality Models

As usual in life sciences, we assume that ontologies and instance sources are versioned so that a specific version reflects a stable data snapshot from a specific point in time. The versioning scheme is assumed to be linear, i.e., a particular version v_i has exactly one successor version v_{i+1} and one predecessor version v_{i-1} . The latest (first) version form exceptions since no successor (predecessor) versions are available.

As illustrated in Figure 2, annotation mappings interrelate a specific version of an instance source with a specific version of an ontology. Furthermore, annotation mappings can refer to common quality taxonomies to specify the quality of individual annotation associations by different criteria, e.g., provenance or stability. Before we define the details of annotation mappings we briefly introduce our models for instance sources and ontologies which are based on [10].

An instance source of version v is denoted by $I_v = (I, t)$ consisting of a set of instances $I = \{i_1, \dots, i_n\}$ and a release timestamp t . An instance item i of I is described by a set of attributes, e.g., name or current status. A special attribute called *accession number* identifies instance items unambiguously. Accession numbers are utilized to reference instance items within annotation mappings.

An ontology $ON_v = (C, R, t)$ of version number v and release timestamp t consists of concepts $C = \{c_1, \dots, c_n\}$ and relationships $R = \{r_1, \dots, r_m\}$. A concept $c \in C$

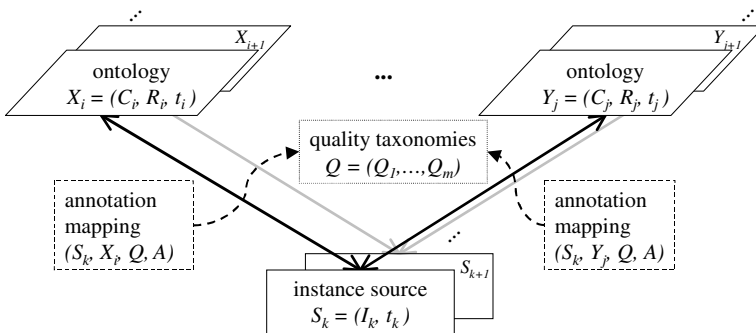


Fig. 2. Model of instance sources, ontologies and annotation mappings with versioning and quality

comprises attributes for its detailed description, e.g., synonyms or a definition. An *accession number* is utilized for unambiguous identification of concepts and the *obsolete* status signals whether a concept is active or not within the ontology. Furthermore, concepts can be interconnected by directed relationships $r = (c_1, c_2) \in R$, e.g., is-a or part-of relationships. Overall, concepts C and relationships R form the graph structure of an ontology which is usually a directed acyclic graph (DAG) with *root concepts* (concepts of C that have no relationships to a super concept).

An annotation mapping $AM = (I_u, ON_v, Q, A)$ associates an instance source version I_u with an ontology version ON_v by a set of correspondences A . A single association or annotation $a \in A$ is denoted by $a = (i, c, \{q\})$, i.e., an instance item $i \in I_u$ is annotated with an ontology concept $c \in ON_v$ and a set of quality indicators (ratings) $\{q\}$. The quality indicators $\{q\}$ of annotations may be numerical values or come from predefined *quality taxonomies* $Q_1, \dots, Q_m \in Q$. Quality taxonomies represent predefined criteria for uniform quality characterization, e.g., the evidence codes for provenance information or stability indicators. Note that for each quality taxonomy at most one quality indicator can be utilized in an annotation. Typically, the quality ratings of an annotation are specified when an annotation is first generated. However annotation ratings may be modified, as seen in the examples of Figure 1, e.g., when changed information about the annotation becomes available.

A quality taxonomy representing a particular quality criterion consists of a set of predefined quality terms $\{q_1, \dots, q_n\}$ which may be arranged in an is-a-like hierarchy. In the general case, a quality term $q = (q', type)$ of name q is defined by a *type* and an optional super term q' . Every quality term has exactly one parent term, if no parent term exists, the quality term is assumed to be the root of the quality taxonomy. Quality terms can be of two different types: *instantiable* and *abstract*. While instantiable quality terms are applicable for rating an annotation, abstract ones are not utilized in annotations, i.e., they only act as aggregation nodes within the taxonomy. For our study, we assume that quality taxonomies remain unchanged.

We will utilize three different types of quality indicators to specify (1) *provenance type*, (2) *stability* and (3) *age* of annotations. First, for provenance information we utilize and analyze the existing Evidence Codes (EC) [8] for GO annotations which specify their generation method. Figure 3 shows the current *EC quality taxonomy* including different groups, in particular ‘Manually assigned’ (*man*), ‘Automatically assigned’ (*auto*) and ‘Obsolete’ (*obs*). Manually determined annotations are further

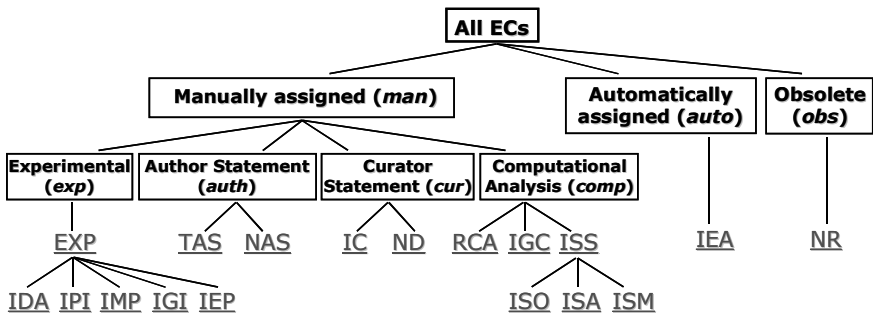


Fig. 3. Evidence Code Taxonomy

refined by the *exp*, *auth*, *cur* and *comp* groups. In contrast, *auto* annotations are unverified but have been generated by algorithms such as homology or keyword mappings. For stability and age, we do not directly use numerical values but map them into categorical terms of a quality taxonomy to simplify their use and evaluation. Our *stability quality taxonomy* consists of only two terms to differentiate *stable* and *unstable* annotations based on their evolution history. Our *age quality taxonomy* differentiates between *novel*, *middle* and *old* annotations. Hence, an automatically generated, stable and middle-aged annotation between instance item i and ontology concept c can be described by $a = (i, c, \{IEA, stable, middle\})$. The introduced quality taxonomies will be used in our evaluation in Section 4. Note that the EC information is frequently available for GO annotations but has not yet been comparatively evaluated. Furthermore, to the best of our knowledge the stability and age of annotations has not yet been analyzed and utilized.

In life sciences, annotation mapping versioning usually follows the versioning scheme of the instance source, i.e., a new instance source version possibly includes changed annotations as well as referring to some (current or older) versions of the respective ontologies. On the other hand, a new ontology version is generally not released with a new version of annotation mappings. Furthermore, succeeding versions of an instance source may refer to the same ontology version.

2.2 Evolution Model

We extend the evolution model for ontologies and mappings of [10] which is limited to simple addition and deletion changes. In order to study evolution in annotations in more detail, we introduce new change types and consider quality changes in annotations as well as the influence of instance / ontology changes on annotations.

Figure 4 summarizes the possible change operations for instances, ontologies and annotations in a simple taxonomy. For instance sources (object \triangleq instance item) and ontologies (object \triangleq ontology concept), we distinguish between the following operations:

- *add*: addition of a new object
- *del*: deletion of an existing object
- *toObs*: marking an existing object as obsolete, i.e., the object becomes inactive
- *subs*: substitution of an existing object by a new object
- *merge*: merging of an object into an existing object

For annotations we differentiate between the following change operations based on the operations for instance sources and ontologies:

- *add*: addition of a new annotation
- *del_{ann}*: deletion of an existing annotation
- *del_{ont}*: deletion of an annotation caused by ontology concept change or delete
- *del_{ins}*: deletion of an annotation caused by instance item change or delete
- *chg_{ont}*: adaptation of an annotation caused by ontology concept change
- *chg_{ins}*: adaptation of an annotation caused by instance item change
- *chg_{qual}*: change of the quality indicator of an annotation

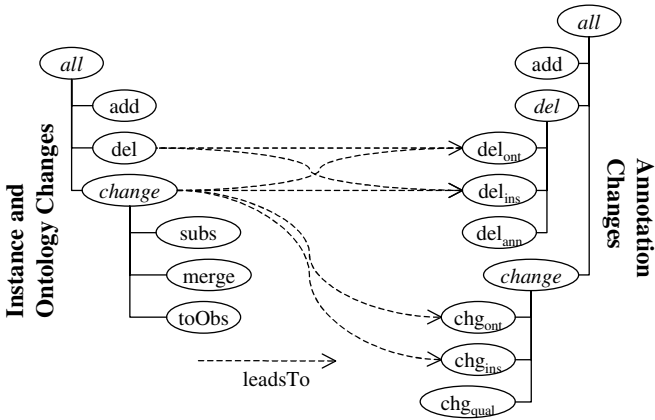


Fig. 4. Effects of instance and ontology changes on annotations

Several dependencies exist between instance/ontology changes and annotation changes (see *leadsTo* dependencies in Figure 4) leading to a corresponding propagation of changes when ontologies and instances evolve. Deletions of ontology concepts and instances always lead to the removal of dependent annotations (*del_{ont}*, *del_{ins}* changes). Furthermore, a change (*subs*, *merge*, *toObs*) of an instance item or ontology concept may cause the deletion or adaptation of dependent annotations as described with the *del_{ins}*, *del_{ont}*, *chg_{ins}* and *chg_{ont}* operations. Besides these changes quality changes (*chg_{qual}*), e.g., when an automatically generated annotation was later proved by an experiment, and conventional additions / deletions (*add*, *del_{ann}*) for annotations are distinguished.

Figure 5 illustrates the various change operators by a rather comprehensive example of annotation evolution. The example displays an evolution step between two versions for an instance source $I (I_1 \rightarrow I_2)$, an ontology $ON (ON_1 \rightarrow ON_2)$ and an annotation mapping $AM ((I_1, ON_1) \rightarrow (I_2, ON_2))$. The table on the left summarizes the change

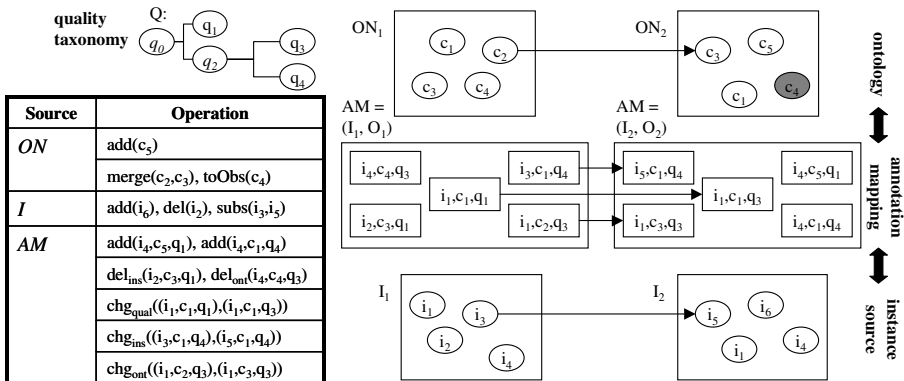


Fig. 5. Evolution example with possible change operations

operations resulting in the new versions for I , ON and AM , shown on the right of Figure 5. So the instance source as well as the ontology possess added (i_6, c_5) and deleted objects (i_2, c_4) . For c_2 a merge into concept c_3 was performed and c_4 has become obsolete. Furthermore, i_3 was replaced by the new instance item i_5 . As a result some annotations were adapted, e.g., (i_1, c_2) to (i_1, c_3) and (i_3, c_1) to (i_5, c_1) , or deleted, e.g., (i_2, c_3) and (i_4, c_4) . Moreover, (i_1, c_1) changed its quality from q_1 to q_3 in the new version. New annotations were also added: (i_4, c_5, q_1) and (i_4, c_1, q_4) .

2.3 Measures to Quantify Annotation Evolution and Changes

For our evaluation, we will utilize several measures to quantitatively assess the evolution of life science annotations. In addition to some general cardinality and growth measures we want to specifically evaluate annotation changes such as the change propagations between instances/ontologies and annotations as well as changes in the quality of annotations.

By using quality-specific statistics we can quantify how annotations with different quality indicators evolve over time, e.g., to discover which quality groups (annotations with a particular quality q) changed heavily or remained almost stable in a period p under review. For these purposes, we use the following measures:

	$ A_{v_i} $	number of annotations in version v_i of an annotation mapping	
	$ A_{v_i, q} $	number of annotations with quality q in version v_i	
	$ A_{v_i, q} / A_{v_i} $	relative share of annotations with quality q to the overall number of annotations in version v_i	
$Add_{v_i, v_j, q}$	$Del_{v_i, v_j, q}$	$Chg_{v_i, v_j, q}$	number of <i>added</i> , <i>deleted</i> or <i>changed</i> annotations with quality q between version v_i and v_j
$Add_{p, q}$	$Del_{p, q}$	$Chg_{p, q}$	number of <i>added</i> , <i>deleted</i> or <i>changed</i> annotations with quality q within an observation period p
$growth_{A, q, v_i, v_j} = A_{v_j, q} / A_{v_i, q} $			growth rate of annotations with quality q between version v_i and v_j

We further investigate the impact of instance/ontology changes on annotation changes. Since instance/ontology changes especially deletions, merges or substitutions affect changes in annotations we propose measures that assess these influences w.r.t. a version change ($v_i \rightarrow v_j$) or an observatio006E period (p):

Chg_{ont}	Chg_{ins}	number of annotations that have changed caused by a change of the referenced instance item or ontology concept
	Chg_{qual}	number of annotations that changed their quality
Del_{ont}	Del_{ins}	number of annotations that have been deleted caused by a change or a deletion of the referenced instance item or ontology concept

3 Assessment of Annotation Stability

In this section we propose a method to assess the stability of annotations based on their evolution history and changes in quality indicators. To assess the evolution history without considering quality criteria, we define the history h of an annotation $a = (i, c)_n$ of version v_n :

$$h((i, c)_n) = ((i, c)_0, (i, c)_1, \dots, (i, c)_n) \mid 0 \leq i < n: (i, c)_i \rightarrow (i, c)_{i+1}$$

So an annotation $(i,c)_{i+1}$ in v_{i+1} has evolved from $(i,c)_i$ in v_i , e.g., caused by an instance merge or substitution (see change taxonomy in Figure 5), or remained unchanged. The non-existence of an annotation in a version is denoted by a null value, e.g., after a deletion or before the first occurrence. The computation occurs with respect to all versions of a predefined observation period p , e.g., the last year. Given the history h for an annotation a we can determine different measures for its evolution within an observation period p .

First, the age of an annotation (in number of versions) is defined as

- $a_{age} = (n - fo) + 1$

where n is the number of the current version (v_n) and fo denotes the number of the version (v_{fo}) in which the annotation occurs for the first time within p . In addition, we count the number of versions in p in which an annotation appeared ($a_{present}$). Note that the counts ignore all versions of the annotation mapping before the first occurrence of an annotation. Based on a_{age} and $a_{present}$ we define a simple *existence stability* measure that evaluates the relative existence of a single annotation a :

- $stab_{exis}(a) = a_{present} / a_{age}$

To evaluate quality changes of annotations within p we use an extended history h_Q of an annotation with respect to a quality indicator (e.g., provenance):

$$h_Q((i,c,q)_n) = ((i,c,q)_0, (i,c,q)_1, \dots, (i,c,q)_n) \mid 0 \leq i < n: (i,c,q)_i \rightarrow (i,c,q)_{i+1}$$

The extended history h_Q incorporates the values of the considered quality indicator w.r.t. a particular quality taxonomy Q . Note that the consideration of quality changes in an annotation history may only be useful for some quality criteria. For instance, we will focus on provenance changes in our evaluation, e.g., when the evidence code of an annotation is modified due to new experimental findings. We count quality changes by determining the number of versions in the history of a where a quality change occurred ($a_{changed}$). Conversely, $a_{unchanged}$ specifies the number of versions without quality modification. Versions for which an annotation was temporarily missing are skipped in the change comparison of the quality indicator.

Utilizing the counts we define a stability measure for *quality stability* as well as a *combined stability* for a single annotation a :

- $stab_{qual}(a) = a_{unchanged} / (a_{unchanged} + a_{changed})$
- $stab_{comb}(a) = \min(stab_{qual}(a), stab_{exis}(a))$

While $stab_{qual}$ assesses the frequency of quality changes of an annotation, the combined stability measure $stab_{comb}$ conservatively integrates $stab_{exis}$ and $stab_{qual}$ by calculating the minimum. Note that the proposed measures have a value range of $[0,1]$. Thereby, a low value signals instability. Perfect stability is achieved in case of 1, e.g., if an annotation is permanently present since its first occurrence (perfect existence stability) or possesses no quality changes (perfect quality stability). In our evaluation (Section 4) we will utilize these measures to classify annotations w.r.t. the two quality criteria *age* and *stability* discussed in Section 2.1. Particularly, we use a threshold criterion to map numerical stability values into corresponding terms of the stability taxonomy.

q ₁	q ₁	q ₁	q ₁	(i ₁ ,c ₁ ,q ₁)
q ₁	/	/	q ₁	(i ₂ ,c ₂ ,q ₁)
/	q ₂	q ₂	q ₁	(i ₃ ,c ₃ ,q ₃)
/	/	/	q ₂	(i ₄ ,c ₄ ,q ₂)
v ₀	v ₁	v ₂	v ₃	v ₄

\xrightarrow{p}

annotation <i>a</i>	<i>a</i> _{age}	<i>stab</i> _{exis}	<i>stab</i> _{qual}	<i>stab</i> _{comb}
(i ₁ ,c ₁ ,q ₁)	5	5/5 = 1	4/(4+0) = 1	1
(i ₂ ,c ₂ ,q ₁)	5	3/5 = 0.6	2/(2+0) = 1	0.6
(i ₃ ,c ₃ ,q ₃)	4	4/4 = 1	1/(1+2) = 0.33	0.33
(i ₄ ,c ₄ ,q ₂)	2	2/2 = 1	1/(1+0) = 1	1

Fig. 6. History and measure results of four example annotations

The example in Figure 6 illustrates the proposed measures for four annotations. An observation period with 5 versions of an annotation mapping (v_0-v_4) is considered. For each version the quality term of an annotation is displayed, an empty cell denotes the temporal non-existence of an annotation in the respective version. The four histories of (i_1, c_1, q_1) , (i_2, c_2, q_1) , (i_3, c_3, q_3) and (i_4, c_4, q_2) of version v_4 exhibit different evolution characteristics. Annotation (i_1, c_1, q_1) has been introduced in v_0 (i.e., $a_{age}=5$) and shows a perfect stability of 1 in $stab_{exis}$ as well as $stab_{qual}$ and thus also in $stab_{comb}$. By contrast, annotation (i_2, c_2, q_1) of the same age possesses periods of temporal non-existence (v_1, v_2) resulting in a low existence stability of 0.6. Furthermore, (i_3, c_3, q_3) is continuously present in 4 versions of p but received two quality changes ($q_2 \rightarrow q_1 \rightarrow q_3$). Hence, the quality and the combined stability are poor (0.33). The last annotation (i_4, c_4, q_2) shows a perfect combined stability, however it is quite novel ($a_{age}=2$) due to its first occurrence in version v_3 .

4 Evaluation

In our evaluation experiments we comparatively analyze the evolution of annotations in the two large annotation sources Ensembl [11] and Swiss-Prot [3] which annotate their proteins with concepts of the Gene Ontology [9]. We first analyze how the annotations evolved for the different provenance types, i.e., different kinds of evidence codes, and how instance (protein) and ontology changes propagated to annotations. In Section 4.2, we additionally analyze the age and stability indicators of Section 3.

4.1 Provenance Analysis

For our study we use available Swiss-Prot and Ensembl versions between March 2004 and December 2008. During this observation period Swiss-Prot (Ensembl) released 14 (28) major versions, namely versions 43-56 (25-52). Both sources provide many functional protein annotations for various species. Whereas Swiss-Prot primarily contains manually curated entries, Ensembl focuses on the automatic generation and integration of data. We consider the functional annotations of human proteins with the concepts of the Gene Ontology (GO) [9] which consists of the three sub ontologies ‘biological process’, ‘molecular function’ and ‘cellular component’. In the following we do not differentiate between these sub-ontologies and treat GO as one ontology.

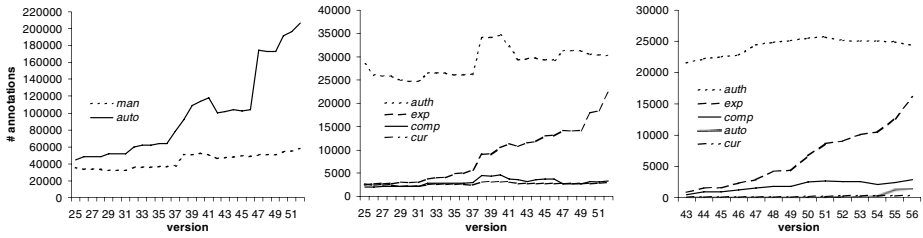


Fig. 7. Evolution of annotations in different EC groups

- Manually curated vs. automatically assigned (Ensembl)
- “Subclasses” of manually-curated (Ensembl)
- All annotations (Swiss-Prot)

Note that Swiss-Prot always attempts to incorporate the current GO release whereas Ensembl often relies on older GO releases in several versions.

Figure 7 shows how the number of GO annotations evolved for different evidence code groups of the EC taxonomy for Ensembl and Swiss-Prot, respectively. Figure 7(a) indicates that Ensembl is dominated by automatically assigned GO annotations (about 78% of the 265,000 annotations in the last version). Furthermore, the growth in the number of automatically determined annotations is very high (factor 4.6 within the last four years). In addition, there is a substantial number of deletions between v_{40} to v_{42} . By contrast, the manually curated annotations grew only modestly by a factor of 1.7. Figure 7(b) shows the development for the manually determined annotations in more detail. We observe a strong increase for experimentally validated annotations ($growth_{exp}$: 8.9) while author statement annotations increased only slightly ($growth_{auth}$: 1.1). The number of curator and computational assigned annotations remained on a very low level.

Figure 7(c) illustrates the evolution of annotations in Swiss-Prot which currently covers about 45,000 annotations, i.e., about six times less than Ensembl. In contrast to Ensembl, Swiss-Prot contains very few automatically generated annotations (1,440) which were recently introduced. The main part of Swiss-Prot annotations encompasses *auth* annotations (about 24,000 in v_{56}). Note that their number is slightly decreasing since v_{51} . The number of *exp* annotations has significantly increased ($growth_{exp}$: 18.5) to about 16,000 at present. Overall, Swiss-Prot provides predominantly manually curated annotations that exhibit a continuous, stable evolution without remarkable fluctuations.

The table in Figure 8 summarizes the number of evolution operations that have been carried out since March 2004 in Swiss-Prot and Ensembl. To determine the changes we compared objects of different versions based on their accession numbers to generate sets of added or deleted objects. More complex changes such as the substitution or merge of proteins that may cause annotation changes (Chg_{ins}) or deletions (Del_{ins}) were identified with the help of evolution information provided by the source distributors. Particularly, Swiss-Prot offers web services to keep track of the protein history, e.g., accession number changes, while Ensembl logs change events between released versions, e.g., what proteins were replaced by others in a new version. Whereas the

		<i>Add</i>	<i>Chg</i>			<i>Del</i>		
			<i>Chg_{ins}</i>	<i>Chg_{ont}</i>	<i>Chg_{qual}</i>	<i>Del_{ann}</i>	<i>Del_{ins}</i>	<i>Del_{ont}</i>
S	abs. (%)	32,613 (53%)	18,214 (30%)			10,502 (17%)		
		-	16,106	56	2,052	8,511	1,369	622
E	abs. (%)	391,771 (60%)	47,805 (8%)			208,585 (32%)		
		-	4,310	171	43,324	145,209	60,788	2,588

Fig. 8. Number (and percentage) of evolution operations aggregated over all versions in Swiss-Prot (Sp) and Ensembl (E)

	Swiss-Prot						Ensembl					
	<i>Add</i>		<i>Chg</i>		<i>Del</i>		<i>Add</i>		<i>Chg</i>		<i>Del</i>	
<i>exp</i>	15,751	48.2%	1,830	10.0%	1,784	17.0%	25,979	6.6%	5,826	12.2%	7,575	3.6%
<i>auth</i>	11,307	34.6%	15,177	83.3%	7,350	70.0%	34,046	8.7%	16,381	34.3%	29,148	14.0%
<i>cur</i>	339	1.0%	65	0.4%	73	0.7%	6,362	1.6%	300	0.6%	6,318	3.0%
<i>comp</i>	3,730	11.4%	1,107	6.1%	1,214	11.6%	6,734	1.7%	5,720	12.0%	4,362	2.1%
<i>auto</i>	1,541	4.7%	35	0.2%	81	0.8%	316,979	80.9%	18,344	38.4%	157,632	75.6%
<i>obs</i>	0	0.0%	0	0.0%	0	0.0%	1,826	0.5%	1,234	2.6%	3,550	1.7%
sum	32,668		18,214		10,502		391,926		47,805		208,585	

Fig. 9. Distribution of the operations add, change, delete in different EC groups in Ensembl and Swiss-Prot

majority of changes are additions (60% in Ensembl, 53% in Swiss-Prot) there is a surprising number of deletions and changes, apparently influenced by some major reorganization such as introduction of new accession numbers. For example, in Swiss-Prot about 30% of all evolution changes are annotation changes (*Chg*) which were primarily caused by instance changes keeping corresponding annotations alive instead of deleting them. By contrast, annotation changes in Ensembl are dominated by quality (here: EC code) changes. In both sources ontology changes only marginally influence changes on annotations. This is also influenced by the fact that annotations are administered within the instance sources while ontologies are developed independently from the instances. Finally, the number of deletions is non-negligible in both sources especially in Ensembl where 32% of all changes are annotation deletions.

We now analyze the distribution of the evolution operations *add*, *change* and *delete* for the different EC groups, as summarized in Figure 9. In Swiss-Prot about one half of the additions are experimentally validated annotations and a third comprises *auth* annotations. By contrast, change (83%) and delete operations (70%) primarily occur for *auth* annotations indicating a rather high instability for this provenance type. On the other hand, Ensembl predominantly adds and deletes automatically generated annotations (81% and 75% of all adds/deletes, respectively). Annotation changes are distributed mainly over automatically assigned (38%) and author statement annotations (34%). In summary, the evolution of existing annotations occurs primarily for *auto* and *auth* annotations.

We further analyze provenance (EC) changes in more detail to see which new EC codes are chosen for improved annotation quality. The tables in Figure 10 aggregates EC changes in Swiss-Prot and Ensembl for versions since March 2004. Each cell

from / to	exp	auth	cur	comp	auto	Sum		from / to	exp	auth	cur	comp	auto	obs	Sum	
exp	147	24	0	42	1	214	10%	exp	896	413	11	1,259	2,966	3	5,548	13%
auth	1,121	270	34	165	0	1,590	72%	auth	1,592	798	73	1,038	11,901	23	15,425	35%
cur	7	9	0	3	0	19	1%	cur	21	27	0	16	182	0	246	1%
comp	160	197	7	0	0	364	16%	comp	1,280	1,206	26	0	3,101	0	5,613	13%
auto	16	4	0	1	0	21	1%	auto	3,311	10,169	228	2,329	0	116	16,153	37%
Sum	1,451	504	41	211	1	2,208		obs	79	391	9	12	725	0	1,216	3%
	66%	23%	2%	10%	0%			Sum	7,179	13,004	347	4,654	18,875	142	44,201	
									16%	29%	1%	11%	43%	0%		

Fig. 10. Evidence codes changes in Swiss-Prot (left) and Ensembl (right)

outlines how many annotations changed *from* one evidence code (rows) *to* another (columns). Note, that we aggregate changes into the EC groups *exp*, *auth*, *cur*, *comp*, *auto* and *obs*, e.g., changes from ISS to TAS are summarized in “from *comp* to *auth*” while changes from IPI to IDA are mapped into “from *exp* to *exp*”. We observe that, annotation changes in Swiss-Prot primarily (72%) occur for author statement (*auth*) annotations and that most new annotations (66%) are experimentally proved (*exp*). This shows the progress of annotation development in the recent years by increasingly using biologically proved annotations which are preferred over mere author statements. In Ensembl, the vast amount of automatically generated annotations leads to a somewhat different picture. Only for the shares of two EC groups, *auto* and *exp*, there is an increase for the new EC codes compared to the original ones. All other EC types reduced their shares due to EC changes, especially *auth* annotations. Most EC changes occurred – in both directions – between *auto* and *auth* annotations indicating a high instability of these provenance categories.

4.2 Age and Stability Analysis

In addition to the evidence code (provenance) information, we now analyze the age and stability measures introduced in Section 3. This analysis occurs for the currently available annotations in the latest versions of Ensembl and Swiss-Prot. We compare these annotations with all versions in the last three years (p), i.e., we use the versions 26-52 of Ensembl and versions 47-56 of Swiss-Prot.

We map the age and stability values into quality taxonomies mentioned in Section 2. We differentiate three *age* groups: annotations that exist since half a year (*novel*), those that were generated between half and one and a half years ago (*middle*) and annotations that are older than one and a half years (*old*). For the stability criteria $stab_{exists}$, $stab_{qual}$ and the combination $stab_{comb}$ we use a minimum threshold of 0.9 for *stable* annotations; lower values indicate *unstable* annotations. Hence, a stable annotation must be present in at least 90% of the versions since its first occurrence and at most 10% quality (EC) changes can occur in the history of an annotation. Note, that we leave out all annotations with evidence code NR (*not recorded*) and ND (*no biological data available*) since these annotations provide no valuable information.

Figure 11 displays the classification results of our method for both annotation sources. The 45,000 (263,000) annotations in Swiss-Prot (Ensembl) are classified using the three mentioned criteria: *provenance* (rows), *age* (columns) further separated by the three stability criteria. White (grey) rows denote the number of

	<i>old</i>			<i>middle</i>			<i>novel</i>			
	<i>lstab_{exis}</i>	<i>lstab_{qual}</i>	<i>lstab_{comb}</i>	<i>lstab_{exis}</i>	<i>lstab_{qual}</i>	<i>lstab_{comb}</i>	<i>lstab_{exis}</i>	<i>lstab_{qual}</i>	<i>lstab_{comb}</i>	
Swiss-Prot	<i>exp</i>	7,980	6,965	6,905	2,306	2,266	2,266	5,655	5,637	5,637
		84	1,099	1,159	0	40	40	0	18	18
	<i>auth</i>	22,064	21,913	21,760	1,107	1,101	1,101	1,054	1,054	1,054
		169	320	473	0	6	6	0	0	0
	<i>cur</i>	184	160	160	36	36	36	115	115	115
		0	24	24	0	0	0	0	0	0
	<i>comp</i>	1,651	1,599	1,589	364	362	362	845	844	844
		16	68	78	0	2	2	0	1	1
	<i>auto</i>	96	96	96	35	35	35	1,308	1,308	1,308
		1	1	1	0	0	0	0	0	0
sum	31,975	30,733	30,510	3,848	3,800	3,800	8,977	8,958	8,958	
	270	1,512	1,735	0	48	48	0	19	19	
Ensembl	<i>exp</i>	9,473	8,774	8,415	3,378	3,062	3,057	8,808	8,650	8,650
		64	1,340	1,699	9	325	330	0	215	158
	<i>auth</i>	22,421	20,488	19,700	4,244	3,949	3,942	2,492	2,425	2,425
		1,024	2,957	3,745	9	124	311	0	35	67
	<i>cur</i>	238	190	184	67	60	60	157	149	149
		15	63	69	0	7	7	0	8	8
	<i>comp</i>	1,715	1,170	1,079	470	354	353	942	885	885
		198	743	834	7	303	124	0	32	57
	<i>auto</i>	71,082	89,115	68,440	62,136	63,245	61,442	49,909	49,608	49,608
		21,392	3,359	24,034	1,818	709	2,512	0	301	301
sum	104,929	119,737	97,818	70,295	70,670	68,854	62,308	61,717	61,717	
	23,270	8,462	30,381	1,843	1,468	3,284	0	591	591	

Fig. 11. Classification of annotations in Swiss-Prot and Ensembl by provenance, age and stability; $stab > 0.9$ (white), $stab \leq 0.9$ (grey)

annotations that lie above (beyond) the stability threshold. Swiss-Prot covers proportionately more older annotations (72%) than Ensembl (49%). By contrast, the use of automatic annotations allows Ensembl a relative high share (24%) of young/novel annotations. Despite the high share of older annotations, only 4% of the Swiss-Prot annotations are classified as *unstable* compared to 13% in Ensembl (using $stab_{comb}$). In other words, Swiss-Prot (Ensembl) covers 96% (87%) stable annotations.

Considering the three stability criteria one can recognize for both sources that novel and middle aged annotations are rarely classified as *unstable* due to their short history compared to old annotations. Hence, we examine *old* annotations more precisely w.r.t. their stability. In Swiss-Prot the majority of unstable annotations is due to EC changes ($stab_{qual}$, $stab_{comb}$) while relatively few annotations had an existence instability. Most of the existentially unstable annotations ($stab_{exis}$) are of type *auth* while the absolute majority of unstable Swiss-Prot annotations are of type *exp*. This is in accordance to our observations for EC changes (Figure 10) where many annotations changed to experimental proved annotations. Such instabilities for the current annotations may thus be seen as a provenance improvement. In Ensembl the number of

unstable annotations is primarily caused by existential instability (*stab_{exis}*) caused by temporal non-existence of annotations. The majority of unstable annotations occurs for *auto* (79%) and *auth* (12%) annotations confirming their high instability observed earlier.

Our assessment approach seems especially valuable for annotation sources such as Ensembl containing many unverified annotations that are automatically generated. The approach allows the identification of reliable and less reliable annotations w.r.t. three significant criteria: age, provenance and stability. The used measures *stab_{exis}* and *stab_{qual}* constitute orthogonal methods providing different classification results. Users can thus filter a set of annotations, e.g., using only those annotations that existed for a longer time, are experimentally proved or do not show existence or provenance instabilities. For example, one may consider annotations as reliable if they are stable with a middle or old age exhibiting a manual provenance. For these criteria, 34,179 (36,790) annotations of Swiss-Prot (Ensembl) would qualify, i.e., 76% (14%) of all available annotations. Naturally, the selection of quality criteria and the corresponding thresholds (e.g., for age or stability) are highly dependent on the application. So users could also be interested in novel or unstable annotations as these are under strong revision due to a high research interest.

The last aspect underlines that annotation instability is not necessarily a negative feature but may indicate interesting objects or significant new biological findings. Conversely, a high stability may be observed for objects of little interest. The proposed evaluation method allows the selection of either stable or unstable annotations and can thus meet the requirements of different applications and annotation use cases.

5 Related Work

Our work is related to the areas of ontology-based *data quality* and *change management* which have received only little attention so far. The current work on change management mainly focuses on ontologies instead of annotations. There are several approaches that investigate ontology versioning [14,15], define change operations describing differences between two ontologies [17], and formalize the evolution process [20,21]. Complementary, there are only few approaches analyzing the ontology evolution quantitatively [10,24]. In [10] we utilized a generic framework to study the evolution of existing ontologies and to quantify changes of annotation and ontology mappings. Our approach in this paper refines the proposed framework by capturing causes of mapping changes. Hence, we can quantify the changes that have been influenced by ontology and instance changes (additions and deletions) and those resulting from provenance changes, whereas [10] only quantifies added and deleted mapping correspondences (annotations). Furthermore, we introduce and analyze several quality indicators of annotation in this paper.

Data or information quality [19] has been primarily addressed in the context of data integration [16,18]. In life sciences, the quality of annotations especially Gene Ontology annotations including evidence codes has been studied in [6,12,22]. Particularly, the case study in [6] assesses annotation quality by using quality-scores for ECs thereby the scores are intuitively defined by the authors. They show descriptive and comparative statistics w.r.t. the quality-scores and annotations in model eukaryotes.

Furthermore, [12] developed a method to estimate the error rate of curated sequence annotations for a particular evidence code (ISS). The approach utilizes the GOSeqLite database to compare annotations that were generated by sequence similarity vs. those that were not. In [22] the authors recommend the utilization of ECs as an indicator for their reliability. In addition, they show simple distribution statistics of annotations for three self-defined classes (homology-based, literature-based and others) and different species but do not examine the annotation evolution. In contrast to previous work on annotation quality, we propose a generic evolution model allowing a multidimensional analysis of annotations w.r.t. different quality taxonomies (age, stability, provenance). The model makes heavily use of quantified evolutionary changes on instance and ontology level but also includes annotation (quality) modifications.

Like our work, [23] provides stability measures to rate correspondences of available mappings but is focused on ontology mappings interconnecting two ontologies. The idea behind this approach is to consider the correspondence stability in addition to the computed element similarity. Conversely, our approach in this paper focuses on annotation mappings and takes multiple quality taxonomies into account to specify and classify the quality of annotations.

6 Conclusion and Future Work

We propose a generic approach to estimate the quality of ontology-based annotations by taking their evolution history into account. The approach considers instance and ontology changes and their influence on annotation mappings. Our annotation model supports different quality measures, such as provenance, age, and stability and the use of quality taxonomies. For provenance information we utilize existent information on evidence codes. We propose different stability measures for annotations taking temporal non-existence and provenance changes into account. Our approach can be used in different scenarios, e.g., by various analysis applications to filter ingoing annotations and by annotation providers to improve their data quality, especially when they integrate annotations from other data sources.

We applied our model-based approach in a comparative evaluation to study functional protein annotations provided by two large life science annotation sources, namely Swiss-Prot and Ensembl. We observed that most annotation changes are additions of new annotations but there are also many changes and deletions of existing annotations. Most of the annotation changes are caused by instance changes or evidence code changes while ontology changes had a minor impact on existing annotations. We also observed that new experimental findings frequently cause the evidence code of existing annotations to be updated. The high instability was observed for automatically generated annotations (in Ensembl) and annotations based on author statements.

We see several directions for future work. First, our annotation model can be applied for additional annotation data sets, e.g., for different species. Second, the proposed approach can be utilized for enhancing instance-based matching techniques that heavily depend on the reliability of input annotations. Likewise, the quality of automatically generated annotations can probably be improved when they are based on existing high quality annotations, e.g., to avoid verified annotations to be overwritten by automatically determined ones or to mark them as new when they are generated for the first time.

Acknowledgements. This work was supported by BMBF grant 01AK803E “Medi-GRID – Networked Computing Resources For Biomedical Research” as well as the Interdisciplinary Centre for Bioinformatics founded by the German Research Foundation (DFG).

References

1. Berriz, G.F., King, O.D., Bryant, B., et al.: Characterizing gene sets with FuncAssociate. *Bioinformatics* 19(18), 2502–2504 (2003)
2. Bose, R., Frew, J.: Lineage retrieval for scientific data processing: A survey. *ACM Computing Surveys* 37(1), 1–28 (2005)
3. Boutet, E., Lieberherr, D., Tognolli, M.: UniProtKB/Swiss-Prot. *Methods in Molecular Biology* 406, 89–112 (2007)
4. Boyle, E.I., Weng, S., Gollub, J., et al.: GO:TermFinder - open source software for accessing Gene Ontology information and finding significantly enriched Gene Ontology terms associated with a list of genes. *Bioinformatics* 20(18), 3710–3715 (2004)
5. Buneman, P., Chapman, A., Cheney, J.: Provenance management in curated databases. In: Proc. of the 2006 ACM SIGMOD International Conference on Management of Data, pp. 539–550 (2006)
6. Buza, T.J., McCarty, F.M., Wang, N.: Gene Ontology annotation quality analysis in model eukaryotes. *Nucleic Acids Research* 36(2), e12 (2008)
7. Dahlquist, K.D., Salomonis, N., Vranizan, K., et al.: GenMAPP, a new tool for viewing and analyzing microarray data on biological pathways. *Nature Genetics* 31(1), 19–20 (2002)
8. Gene Ontology - Evidence Codes:
<http://www.geneontology.org/GO.evidence>
9. The Gene Ontology Consortium: The Gene Ontology project in 2008. *Nucleic Acids Research* 36, D440–D441 (2008) (Database issue)
10. Hartung, M., Kirsten, T., Rahm, E.: Analyzing the Evolution of Life Science Ontologies and Mappings. In: Bairoch, A., Cohen-Boulakia, S., Froidevaux, C. (eds.) DILS 2008. LNCS (LNBI), vol. 5109, pp. 11–27. Springer, Heidelberg (2008)
11. Hubbard, T.J., Aken, B.L., Ayling, S., et al.: Ensembl 2009. *Nucleic Acids Research* 37, D690–D697 (2009) (Database issue)
12. Jones, C.E., Brown, A.L., Baumann, U.: Estimating the annotation error rate of curated GO database sequence annotations. *BMC Bioinformatics* 8(1), 170 (2007)
13. Kirsten, T., Thor, A., Rahm, E.: Instance-based matching of large life science ontologies. In: Cohen-Boulakia, S., Tannen, V. (eds.) DILS 2007. LNCS (LNBI), vol. 4544, pp. 172–187. Springer, Heidelberg (2007)
14. Klein, M.: Change Management for Distributed Ontologies. PhD thesis, Vrije Universiteit Amsterdam (2004)
15. Klein, M., Fensel, D.: Ontology versioning on the Semantic Web. In: Proceedings of the International Semantic Web Working Symposium (SWWS), pp. 75–91 (2001)
16. Naumann, F., Leser, U., Freytag, J.C.: Quality-driven Integration of Heterogeneous Information Systems. In: Proc. of the International Conference on Very Large Data Bases (VLDB), pp. 447–458 (1999)
17. Noy, N., Klein, M.: Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems* 6(4), 428–440 (2004)

18. Rahm, E., Do, H.H.: Data Cleaning: Problems and Current Approaches. *IEEE Data Engineering Bulletin* 23(4), 3–13 (2000)
19. Redman, T.C.: *Data Quality for the Information Age*. Artech House (1996)
20. Stojanovic, L., Maedche, A., Motik, B., Stojanovic, N.: User-driven ontology evolution management. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) *EKAW 2002*. LNCS, vol. 2473, pp. 285–300. Springer, Heidelberg (2002)
21. Stojanovic, L., Motik, B.: Ontology evolution within ontology editors. In: *Proceedings of the International Workshop on Evaluation of Ontology-based Tools*, pp. 53–62 (2002)
22. Thomas, P.D., Mi, H., Lewis, S.: Ontology annotation: mapping genomic regions to biological function. *Current Opinion in Chemical Biology* 11(1), 4–11 (2007)
23. Thor, A., Hartung, M., Gross, A., Kirsten, T., Rahm, E.: An evolution-based approach for assessing ontology mappings - A case study in the life sciences. In: *Proc. Conference of the Business, Technology and Web (BTW)*, pp. 277–286 (2009)
24. Yang, Z., Zhang, D., Ye, C.: Ontology Analysis on Complexity and Evolution Based on Conceptual Model. In: Leser, U., Naumann, F., Eckman, B. (eds.) *DILS 2006*. LNCS (LNBI), vol. 4075, pp. 216–223. Springer, Heidelberg (2006)

Integration and Mining of Genomic Annotations: Experiences and Perspectives in GFINDER Data Warehousing

Marco Masseroli, Stefano Ceri, and Alessandro Campi

Dipartimento di Elettronica e Informazione, Politecnico di Milano,
Piazza Leonardo da Vinci 32, 20133 Milano, Italy
{masseroli,ceri,campi}@elet.polimi.it

Abstract. Many tasks in bioinformatics require the comprehensive evaluation of different types of data, generally available in distributed and heterogeneous data sources. Several approaches, including federated databases, multi databases and mediator based systems, have been proposed to integrate data from multiple sources. Yet, data warehousing seems to be the most adequate when numerous data need to be integrated, efficiently processed, and mined comprehensively. To support biological interpretation of high-throughput gene lists, we previously developed GFINDER (Genome Functional INtegrated Discoverer, <http://www.bioinformatics.polimi.it/GFINDER/>), a web server that statistically analyzes and mines functional and phenotypic gene annotations sparsely available in numerous databanks to highlight annotation categories significantly enriched or depleted in the considered gene lists. GFINDER includes a data warehouse that integrates gene and protein annotations of several organisms expressed through various controlled terminologies and ontologies. Here, we describe GFINDER data warehouse and discuss the lessons learned in its construction and five-year maintenance and development.

Keywords: GFINDER, biomolecular annotation integration, data warehousing.

1 Introduction

In bioinformatics and often in biomolecular research in general, many questions can be addressed only by analyzing different types of data comprehensively, in order to collect enough evidences to support obtained results. As an example, identification of the biomolecular phenomena involved in a specific biological condition usually requires evaluating several structural, functional and phenotypic annotations of the genes resulted differentially expressed in a high-throughput gene expression experiment testing the biological condition. Publicly accessible molecular biology databanks providing such gene annotations are continuously increasing in number (more than 1,150 in January 2009 [1]) and in coverage of the included biomolecular entities (e.g., genomic DNAs, genes, transcripts, proteins), as well as of their described structural and functional biomedical features (e.g., nucleotide or amino acid sequences; expression in different tissues; involvement in biological processes and

genetic disorders). Such databanks provide extremely valuable information to support the interpretation of experimental results (e.g., high-throughput lists of candidate relevant genes or proteins) and infer new biomedical knowledge. Yet, the information about a given biomolecular entity is often scattered across many different databanks: the nucleotide sequence of a gene may be stored in one databank, information about the expression of the gene may be stored in a second databank, the three-dimensional structure of its products may be stored in a third one, and data regarding interactions of the gene products with other proteins may be stored in a fourth databank. Combining information from multiple databanks is hence paramount for biomolecular investigation. Moreover, since different biomolecular databanks often contain redundant or overlapping information, integration of data they provide allows cross-validation of the available data in order to identify mismatching information.

Several approaches, including data warehousing (e.g., Biowarehouse [2]), multi databases (e.g., TAMBIS [3]), federated databases (e.g., DiscoveryLink [4]), information linkage (e.g., SOURCE [5]) and mediator based solutions (e.g., Biomediator [6]), have been proposed to integrate data from multiple sources. Yet, data warehousing seems to be the most adequate when the data to be integrated are numerous and off-line processing is required to efficiently and comprehensively mine the integrated data. This approach requires that information from the distributed databanks to be integrated are automatically retrieved and processed in order to create and maintain updated an integrated and consistent collection of originally distributed data. Besides several other bioinformatics applications, data warehousing has been used to build the backend of various bioinformatics tools and web systems, including those for gene ID list annotation and enrichment analysis [7], such as GFINDER [8].

2 GFINDER System and Data Warehouse

In order to enable performing comprehensive evaluations of functional and phenotypic gene annotations sparsely available in numerous different databanks accessible via the Internet, we previously developed GFINDER (Genome Functional INtegrated Discoverer, <http://www.bioinformatics.polimi.it/GFINDER/>). It is a web server system that integrates genomic and proteomic annotations of user uploaded gene lists and allows performing their statistical analysis and mining aimed at highlighting those annotation categories that are significantly enriched or depleted in the uploaded gene lists. Towards this aim, GFINDER is implemented in a three-tier architecture (Figure 1) based on a multi-database data warehouse. In such architecture, independent and interconnected modules take advantage of several controlled terminologies and ontologies, which describe gene-related biomolecular processes, functions and phenotypes, stored in the data warehouse together with their associations with genes and proteins of several organisms. In particular, the architecture first tier, the multi-database data tier, is implemented in a MySQL relational DBMS server that manages system users and their data, all different genes, gene products and types of annotations integrated, and the generated data analysis results in three databases. The Master DB database maintains information about the web server users, their uploaded gene lists, and their accesses and navigation paths within the GFINDER system. Another database, MyGO DB, keeps information about the Gene Ontology (GO) structure [9] (i.e., GO terms and relationships between them); its schema resembles the

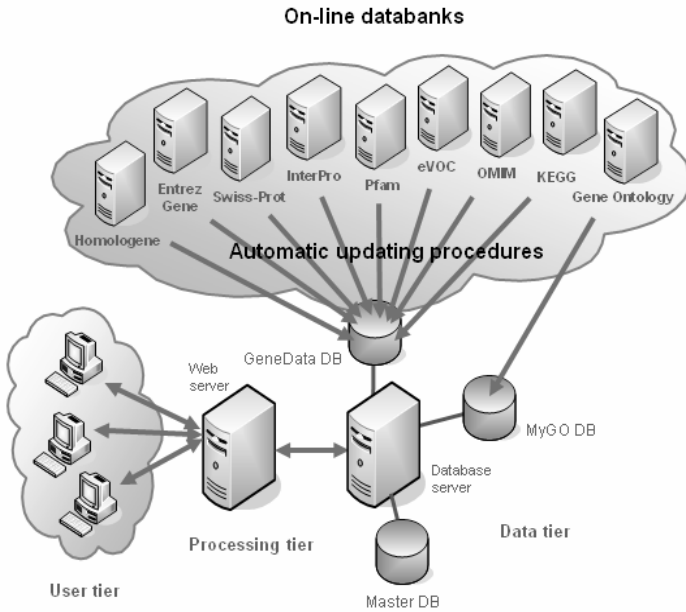


Fig. 1. Three-tier architecture of the GFINDER system

database schema provided by the GO, extended for the inclusion of path-codes and levels of the GO terms within the ontology structure that we calculate on the basis of the GO provided data. A third database, GeneData DB, stores genes and gene products together with their annotations with many different controlled terminologies and ontologies. The conceptual schema of this integrated database was designed according to the approach of modeling individually the different types of data and annotations to be imported in the data warehouse, where such data are integrated by interconnecting all annotations imported from different sources to the genes and proteins they refer to by means of their provided IDs and cross-references.

At time of writing in February 2009, GFINDER data warehouse stored a total of 336,068 genes and 1,125,161 gene products of 14 organisms, including the most important model organisms, retrieved from Entrez Gene, Homologene and UniProt/Swiss-Prot databanks (Table 1), and 15 types of controlled annotations of such genes and proteins retrieved from 6 different biomolecular databanks (Table 2). Controlled annotations in GFINDER data warehouse regard biological processes, molecular functions, cellular components (provided by GO), biochemical pathways (by KEGG), protein families, domains and functional sites (by InterPro and Pfam), genetic disorders, phenotypes and phenotype locations (by OMIM), and descriptions of the gene expression in human anatomical systems, cellular types, developmental stages, and pathologies (by eVOC). Of each controlled terminology and ontology, Table 2 shows the total number of terms and annotations stored in the GFINDER data warehouse for the organisms in Table 1.

Table 1. Organisms and number of their genes and proteins in GFINDER data warehouse at GFINDER opening (3/2004) and after five years (2/2009)

Specie	Genes (3/2004)	Genes (2/2009)	Protein coding genes (2/2009)	Proteins (2/2009)
Baker's yeast (<i>Saccharomyces cerevisiae</i>)	-	6,200	5,880	28,939
Barley (<i>Hordeum vulgare</i>)	-	341	341	447
Clawed frog (<i>Xenopus laevis</i>)	4,091	12,940	12,271	28,804
Cow (<i>Bos taurus</i>)	2,457	26,530	21,004	50,617
Fission yeast (<i>Schizosaccharomyces pombe</i>)	-	5,560	5,069	20,098
Fruit fly (<i>Drosophila melanogaster</i>)	13,851	22,640	14,144	40,970
Human (<i>Homo sapiens</i>)	33,330	40,130	24,682	312,161
Mouse (<i>Mus musculus</i>)	49,718	61,889	34,629	243,334
Nematode (<i>Caenorhabditis elegans</i>)	20,437	21,184	20,186	63,914
Purple Urchin (<i>Strongylocentrotus purpuratus</i>)	318	30,408	28,723	43,130
Rat (<i>Rattus norvegicus</i>)	24,987	36,920	24,746	103,632
Thale cress (<i>Arabidopsis thaliana</i>)	-	33,263	27,217	107,235
Tomato (<i>Lycopersicon esculentum</i>)	-	1,293	1,247	1,924
Zebrafish (<i>Danio rerio</i>)	13,834	36,770	34,940	79,956
TOTAL	163,023	336,068	255,079	1,125,161

Table 2. Controlled terminologies and ontologies and their annotations for the organisms in Table 1 stored in GFINDER data warehouse at GFINDER opening (3/2004) and after five years (2/2009). Terms: elements of a terminology or ontology; Annotated terms: terms actually used to describe the features of the considered organisms; Annotations: term-gene pairs.

Terminology / Ontology	Terms (3/2004)	Terms (2/2009)	Annotated terms (2/2009)	Annotations (2/2009)
GO biological processes	8,172	16,027	7,289	282,289
GO molecular functions	7,280	8,178	3,999	275,099
GO cellular components	1,388	2,288	1,277	220,891
KEGG biochemical pathways	245	396	237	34,462
InterPro protein families	-	11,132	3,885	30,301
InterPro protein domains	-	5,149	2,770	65,569
InterPro protein functional sites	-	1,131	1,016	14,983
Pfam protein domains	2,880	3,484	3,385	55,777
OMIM genetic disorders	1,484	2,465	2,460	3,183
OMIM genetic phenotypes	-	9,912	5,947	17,099
OMIM genetic phenotype locations	-	86	81	5,349
eVOC gene expression in anatomical systems	-	517	164	165,990
eVOC gene expression in cellular types	-	192	56	58,585
eVOC gene expression in developmental stages	-	157	116	90,252
eVOC gene expression in pathologies	-	199	96	75,911
TOTAL	21,449	61,313	32,778	1,395,740

2.1 GFINDER Updater

In order to keep updated the information integrated in the GFINDER data warehouse, which are retrieved from several online databanks frequently updated, we designed and implemented some automatic procedures in Java programming language. They automatically retrieve controlled terminology and ontology information, gene and protein data and their controlled annotations when new releases of them become available in the FTP site of the original databanks. Data files of different format are downloaded from each FTP site and automatically uncompressed, if required. According to the various file formats, specific parsers are used to extract the data of interest from these files and import them into specific GFINDER data warehouse tables. The developed updating procedures have been included in the GFINDER Updater software, which helps maintaining updated the data warehouse monthly and automatically records updating time and amount of data imported for every organism and type of annotation from each databank.

2.2 GFINDER Data Warehouse Releases

At GFINDER system opening in March 2004, GFINDER data warehouse included a total of 163,023 genes of 9 organisms (Table 1) retrieved from Entrez Gene, Homologene and Swiss-Prot databanks, and 6 types of controlled annotations of such genes retrieved from 4 different biomolecular databanks (GO, KEGG, Pfam and OMIM) (Table 2). Since then, GFINDER data warehouse had three major releases. The first, in 2005, regarded the extension of the data warehouse with the integration of gene annotations about genetic phenotypes and phenotype locations taken from the OMIM databank [10]. In the second release, in 2006, protein families, domains and functional sites, provided by InterPro databank together with their annotations to the gene products of the organisms in the data warehouse, were integrated [11]. The third major release, in 2007, regarded the integration of the human gene annotations with four eVOC ontologies describing the gene expression in human anatomical systems, cellular types, developmental stages, and pathologies [12].

Besides the number of controlled terminologies and ontologies, also the amount of their terms and associations with genes and gene products, as well as the quantity of genes and gene products integrated in the data warehouse, continuously increased since GFINDER opening (Table 1 and Table 2). This was due both to the augmented number of organisms included in the GFINDER data warehouse (from 9 to 14), and especially to the increasingly amount of data about such organisms provided by the databanks used as data sources (e.g., from 16,840 to 26,493 total GO terms, and from 22,623 to 42,104 total GO *is a* and *part of* relationships).

Also popularity and usage of GFINDER increased since its opening; in the last year GFINDER web site had about 27,000 accesses from nearly 1,700 distinct IPs, which demonstrate the acknowledgement of the scientific community towards the GFINDER project.

3 Discussion

In the construction and five-year maintenance and development of the GFINDER data warehouse we learned several lessons, which we here discuss with the aim of contributing in defining the bases of a general effective approach for the increasingly needed multi- genome and proteome data warehousing.

As other existing genomic and proteomic data warehouses [2], GFINDER data warehouse was built on a global conceptual schema designed by individually modeling the different types of data and annotations to be integrated as they are provided by the original databanks. Such design allows defining conceptually all considered data types and their attributes and relationships in order to neatly store them without redundancies, and enables to construct queries on the integrated data that can answer complex questions. Yet, when many heterogeneous types of information from different sources need to be integrated in a single data warehouse, as in the usual case of genomic and proteomic data warehouses, it leads to a complex data warehouse schema difficult to be maintained and extended with additional data types. Since usually both these last operations are needed, due to the data structure variations present in subsequent versions of the data to be integrated and the increasing available types of information of interest, such difficulties represent a major limitation for the data warehouse development that should be avoided. Towards this goal, an abstract and generalized conceptual design with vision of possible future expansions, parametric customizable instantiations of the designed entities, and automatic generation of the configured instances should be pursued in the global design and implementation of the data warehouse. Furthermore, as data in the original biomolecular databanks are generally updated frequently, even daily, automatic updating procedures must be implemented at data warehouse creation in order to keep updated its data and quality.

Performance of queries defined to extract information from the data warehouse is also a very important issue in terms of both response time and data mined. Regarding the latter one, one of the approaches implemented in GFINDER is the automatic managing and checking of the biomolecular entity and biomedical feature IDs stored in the data warehouse, which ensures correct use of alias and historical or obsolete IDs. Regarding the former one, i.e. response time of high-throughput queries, besides normalized tables storing the integrated data, in GFINDER data warehouse we also implemented de-normalized data views, designed according to the query requirements, in order to reduce table joins to be repeatedly executed at query time and ensure fast query execution time. Furthermore, some pre-processing of the integrated data is also performed. As an example, unfolding of gene and protein ontological annotations (e.g., GO annotations) is pre-computed and stored to enable faster processing of queries for standard annotation enrichment analysis, which is frequently performed to support interpretation of gene ID lists. These approaches, which consider the specific nature of the data and the queries to be performed on them in addition to standard database theory strategies, should be used to ensure best possible usability performance.

All automatic data processing performed must not only leave unchanged the information content of the imported and integrated data even after their several frequent updates, but it also must be able to find inconsistencies that not rarely exist

between data from different or even the same databank. This is paramount to ensure the best possible quality of data in the data warehouse. Towards this goal, in GFINDER we implemented strict checking of data parsed from source data files. As an example, whenever possible we use regular expression for ID checking and identification, and verify absence of null data and modifications of the data structure in subsequent updating of the source data files. Furthermore, checking and cross-validation of data imported from different databanks should be always performed in order to identify redundant and mismatching information. Analysis of overlaps and relationship loops among imported data can help in verifying data consistency and unveiling unexpected information pattern, and can possibly lead to biological discoveries. As an example, in GFINDER data warehouse, by checking cross references existing between Gene Ontology, Entrez Gene and UniProt databanks, we test consistency of GO annotations of proteins and their codifying genes. By doing so, at time of writing we found that 321 (1.83%) GO annotations (regarding 197 different GO terms) of 81 human proteins were not comprised in the GO annotations of their codifying genes provided by Entrez Gene databank, including also 143 (44.56%) annotations with evidence stronger than that inferred from electronic annotation (IEA). Reporting such inconsistencies to the curators of the databanks providing the inconsistent data can help in improving the quality of the data available in the original databanks to the whole scientific community.

4 Conclusions and Future Developments

Easy and integrated access to the high amount of biomolecular information and knowledge available in many heterogeneous and distributed databanks is required to answer biological questions. Computational systems and data warehouses, such as GFINDER, provide support for comprehensive use and analysis of sparsely available genomic structural, functional and phenotypic information and knowledge. To be effective bioinformatics instruments, biomolecular data warehouses should meet several requirements. In particular, they should have a conceptual schema that ensures the easiest possible maintenance and extensibility of the data warehouse, guarantee good run-time performances of high-throughput queries, and include automatic procedures for updating the integrated data and supporting their quality checking. Ensuring high quality level of the biomolecular information integrated in the data warehouse is an essential requirement in order to enable their subsequent correct analysis and use in support of biological interpretations and knowledge advances. Although quality of the integrated data is always bound to the quality of the data provided by the original databanks, it can be strengthened by automatic correctness checking of all operations performed on the imported data and by testing the accuracy of the imported cross references between data of different databanks. Such data quality controls can lead to reveal inconsistencies or missing information in some public databanks, thus supporting quality improvement of the genomic and proteomic information available to the whole scientific community.

Due to its original design, current conceptual schema of GFINDER data warehouse hampers its extensibility, making difficult the integration of additional types and sources of annotations. To avoid this limitation, we are designing and implementing a

new genomic and proteomic data warehouse that can easily include virtually any annotation data type from any data source thanks to its novel generalized and modular schema. Furthermore, we are redesigning the software architecture that supports creation, extension and automatic update of the data warehouse. The new architecture will automatize as much as possible the plug-in of new data sources making easy the integration of several annotation types from many different biomolecular databanks, as well as the quality checking of the integrated annotations and their structuring in a suitable way to be used for high-throughput data driven biological discoveries.

References

1. Galperin, M.Y., Cochrane, G.R.: Nucleic Acids Research Annual Database Issue and the NAR Online Molecular Biology Database Collection in 2009. *Nucleic Acids Res.* 37(Database issue), D1–D4 (2009)
2. Lee, T.J., Pouliot, Y., Wagner, V., Gupta, P., Stringer-Calvert, D.W., Tenenbaum, J.D., Karp, P.D.: BioWarehouse: A Bioinformatics Database Warehouse Toolkit. *BMC Bioinformatics* 7(170), 1–14 (2006)
3. Stevens, R., Baker, P., Bechhofer, S., Ng, G., Jacoby, A., Paton, N.W., Goble, C.A., Brass, A.: TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources. *Bioinformatics* 16, 184–185 (2000)
4. Haas, L.M., Rice, J.E., Schwarz, P.M., Swops, W.C., Kodali, P., Kotlar, E.: DiscoveryLink: A System for Integrated Access to Life Sciences Data Sources. *IBM Systems Journal* 40, 489–511 (2001)
5. Diehn, M., Sherlock, G., Binkley, G., Jin, H., Matese, J.C., Hernandez-Boussard, T., Rees, C.A., Cherry, J.M., Botstein, D., Brown, P.O., Alizadeh, A.A.: SOURCE: A Unified Genomic Resource of Functional Annotations, Ontologies, and Gene Expression Data. *Nucleic Acids Res.* 31, 219–223 (2003)
6. Cadag, E., Louie, B., Myler, P.J., Tarczy-Hornoch, P.: Biomediator Data Integration and Inference for Functional Annotation of Anonymous Sequences. In: *Pac. Symp. Biocomput.*, pp. 343–354 (2007)
7. Huang, D.W., Sherman, B.T., Lempicki, R.A.: Bioinformatics Enrichment Tools: Paths Toward the Comprehensive Functional Analysis of Large Gene Lists. *Nucleic Acids Res.* 37(1), 1–13 (2009)
8. Masseroli, M., Martucci, D., Pinciroli, F.: GFINDER: Genome Function INtegrated Discoverer through Dynamic Annotation, Statistical Analysis, and Mining. *Nucleic Acids Res.* 32, W293–W300 (2004)
9. The Gene Ontology Consortium: Gene Ontology: Tool for the Unification of Biology. *Nature Genet.* 25, 25–29 (2000)
10. Masseroli, M., Galati, O., Pinciroli, F.: GFINDER: Genetic Disease and Phenotype Location Statistical Analysis and Mining of Dynamically Annotated Gene Lists. *Nucleic Acids Res.* 33, W717–W723 (2005)
11. Masseroli, M., Bellistri, E., Franceschini, A., Pinciroli, F.: Statistical Analysis of Genomic Protein Family and Domain Controlled Annotations for Functional Investigation of Classified Gene Lists. *BMC Bioinformatics* 8(suppl. 1), S14, 1–10 (2007)
12. Ceresa, M., Masseroli, M., Campi, A.: A Web-enabled Database of Human Gene Expression Controlled Annotations for Gene List Functional Evaluation. In: Dittmar, A., Clark, J. (eds.) 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS 2007), pp. 394–397. The Printing House, Stoughton (2007)

Site-Wide Wrapper Induction for Life Science Deep Web Databases

Saqib Mir^{1,2}, Steffen Staab², and Isabel Rojas¹

¹ EML Research

Schloss-Wolfsbrunnenweg 33
D-69118 Heidelberg, Germany

² Institute for Computer Science,
University of Koblenz-Landau,
D-56016 Koblenz, Germany

{saqib.mir, isabel.rojas}@eml-r.villa-bosch.de,
staab@uni-koblenz.de

Abstract. We present a novel approach to automatic information extraction from Deep Web Life Science databases using wrapper induction. Traditional wrapper induction techniques focus on learning wrappers based on examples from one class of Web pages, i.e. from Web pages that are all similar in structure and content. Thereby, traditional wrapper induction targets the understanding of Web pages generated from a database using the same generation template as observed in the example set. However, Life Science Web sites typically contain structurally diverse web pages from multiple classes making the problem more challenging. Furthermore, we observed that such Life Science Web sites do not just provide mere data, but they also tend to provide schema information in terms of data labels – giving further cues for solving the Web site wrapping task. Our solution to this novel challenge of Site-Wide wrapper induction consists of a sequence of steps: 1. classification of similar Web pages into classes, 2. discovery of these classes and 3. wrapper induction for each class. Our approach thus allows us to perform unsupervised information retrieval from across an entire Web site. We test our algorithm against three real-world biochemical deep Web sources and report our preliminary results, which are very promising.

Keywords: Deep Web, Wrapper Generation, Information Extraction, Database.

1 Introduction

Hundreds of freely-accessible databases are available on the Web in the Life Sciences domain, covering areas such as Genomics, Proteomics, Systems Biology and Micro Array Gene Expression, to name a few. These databases often provide complementary data, pertaining to narrow specialized sub-domains. Life Science researchers thus need to search, collect and aggregate data from multiple online resources. This Web site hopping is time consuming and error-prone, whereby a user must learn search

interfaces of various Web sites, perform multiple copy-paste actions, create temporary text-files and manually link records.

“Deep Web” research aims to virtually integrate such Web-accessible databases, provide a unified query interface and, typically, aggregate query results. Deep Web data integration consists of a number of distinct sub-tasks (See [5] for a survey) such as *Source Searching and Clustering* [2, 16, 23], *Interface Extraction* [21, 34], *Interface Matching* [31, 24, 32, 14], *Interface Merging & Query Translation* [15, 19], *Wrapper Generation/Data Extraction* (See section 5 for related work).

We focus on unsupervised wrapper induction and data extraction in this paper. Automatic wrapper induction has received considerable attention in recent years. However, most techniques learn wrappers for one class of Web pages. They assume structurally and content-wise similar pages are manually provided as an input for their wrapper induction methods.

As we shall explain in section 2, in our target domain, data are spread across multiple pages of a Web site, which often differ considerably in their structure and layout (template) as well as content. We therefore need an approach to automatically group similar pages in a Web site for our wrapper induction process. Additionally, we need to automatically discover the Web-site structure, so that we may predict which wrapper to use for a Web page encountered in that Web site.

These requirements go beyond traditional wrapper induction methods. We term this compound problem of Web-page classification, site-structure discovery and wrapper induction as *Site-Wide Wrapper Induction*. Though this task is extremely hard in general, in our target domain, i.e. Web-accessible Life Science databases, we may benefit from additional cues in terms of labeling of data. Consequently, we restrict our attention to Web pages from Web sites with labeled data.

In order to solve the challenge of Site-Wide Wrapper Induction in our target domain, we provide the following original contributions in this paper:

1. A novel approach for unsupervised wrapper induction to extract labeled data
2. An original approach for Web-page classification and site structure discovery
3. An automatic mechanism for detecting and correcting errors in our wrapper learning process

The rest of this paper is organized as follows. Section 2 makes some observations about deep Web scientific sources. Based on these observations, we formulate our problem and present our approach for site-wide wrapper induction in section 3. Section 4 presents our results, followed by a review of the related work and comparison to our contributions in section 5. We conclude the paper in section 6.

2 Online Life Science Databases: Observations and Implications

We observe the following about result pages of Life Science Web sources:

I) Structured Data – The results are highly structured. This owes to the fact that the backend relational schemas are very complex, and entities in scientific domains generally have complex relations and associations.

II) Highly Dynamic Page Structures – Data fields that are NULL are often omitted from the results displayed, resulting in pages with widely varying structure. A

wrapper induction method which ties its learning process to the page structure only would require numerous training pages, covering all possibilities of data arrangement. One drawback of such an approach would be the need for a large number of input seed queries to probe the deep Web source. A related observation is that Web pages undergo frequent updates [18]. An approach which circumvents the need to learn the page structure would hence be desirable.

III) Labeled Data – Scientific data require precision and clear annotation. A natural consequence of this observation is that scientific data are labeled and, often, annotated to controlled vocabularies. We carried out a survey¹ of 20 Biochemical Web sites and found that more than 97% of the data fields on these Web sites were labeled. This differs from other domains such as E-commerce, where many data fields are often unlabeled because they have become self-explaining in the public domain (e.g. price, title). This labeling can be exploited to not only help determine data regions, but can also serve as anchors for these data regions, allowing us to disregard the portion of the page which does not contain data. We further observe that labels for the same real-world entity can be different across Web pages of the same source. Finally, labels of real-world entities, such as biological concepts, rarely change, which is beneficial for wrapper maintenance.

IV) Rich Site Structure – Data is scattered across multiple Web pages. This gives such Web sites a comprehensive structure. Therefore, the wrapper must be able to navigate through the result pages to extract data. We also observe that some data fields, together with their labels, reappear on multiple pages. This reoccurrence can be used for mutual reinforcement, to detect and correct errors in the induction process.

V) Web Services – Our final observation is that of Web service API access. While some Life Science databases do provide such APIs, our survey² of 100 online databases showed that only 11 sources provide programmatic access, and even among these, the coverage of the database in some cases is not complete. Therefore, Web pages still remain the primary form of data dissemination. Furthermore, these services have varying granularity and do not provide any semantics in their descriptions, making unsupervised data extraction extremely difficult.

These observations serve to clarify the two broad characteristics of our work: Firstly, at the Web site level, the challenge is to extract data from a number of pages, generated from many templates. This requires determining homogenous clusters of pages having similar templates so that we can induce wrappers for these clusters. Another implicit requirement is that of learning the structure of a Web site through navigational steps. This is essential because our system needs to know which wrapper to apply during data extraction while traversing through the Web site. Secondly, at the page level, the presence of labeled data gives us the opportunity to segment data records and fields based on these labels, and to accommodate the dynamic structure of pages by using these labels to extract data, rather than analyzing and learning the entire Web page structure in a regular expression-like syntax. However, these labels must themselves be identified. Although the vocabulary for labels converges across

¹ Complete survey available at <http://sabiork.villa-bosch.de/labelsurvey.html>

² Databases indexed by the Nucleic Acids Research Journal (<http://www3.oup.co.uk/nar/database/c/>). Complete survey available at <http://sabiork.villa-bosch.de/servicesurvey.html>

different sources in a domain [7], it is not trivial to manually provide a set of possible labels (which can number in their hundreds) to aid in identification of data regions. Therefore, a desirable approach would be to automatically identify the labels.

3 Wrapper Induction

We present our algorithm for page-level wrapper induction in section 3.1. Subsequently, in section 3.2, we describe how this algorithm is used in our site-wide wrapper-induction method. In section 3.3, we discuss a technique to automatically detect and correct certain erroneous results of our induction algorithm in section 3.1.

3.1 Page-Level Wrapper Induction

Our wrapper induction algorithm relies on multiple sample *instance pages* from a *class of pages*. We borrow this terminology from [9], which describes a *class of pages* in a site as a collection of pages which are generated by the same server-side script or program. Different inputs to this script result in different *instance pages*. We clarify this further using Figure 1, our running example, which shows two instance pages of a class of pages³. The wrapper induction algorithm is shown in Figure 2.

We note that, upon querying, the initial response pages generated by a deep Web source belong to the same class⁴. Therefore, we can probe a source with different inputs, and use the resulting initial pages to learn a wrapper, without having to cluster similarly structured pages. In fact, for a given Web site, the site-wide wrapper induction process is bootstrapped by using these initial result pages and learning a wrapper for this class.

Briefly, the algorithm compares text entries on the sample pages and identifies some (possibly not all) data entries among them. These data entries are subsequently used to identify bigger data regions, so that more data entries can be discovered. A label is then selected for each data entry from text entries outside the data regions, based on vicinity. Our approach is based on the DOM⁵ representation of Web pages,

Entry	c00221				Compound			
Name	beta-D-glucose							
Formula	C6H12O6							
Mass	180.0634							
Reaction	R00026	R01520	R01521	R01522	R01600	R01601	R01602	R02187
	R02887	R03256	R04783	R06077	R06092	R06110	R06144	
Enzyme	1.1.1.47	1.1.3.4	1.1.3.5	2.7.1.1				
	2.7.1.2	2.7.1.63	3.1.6.3	3.2.1.21				
	3.2.1.23	3.2.1.85	5.1.3.3					

Entry	c00185				Compound			
Name	Cellobiose; 1-beta-D-Glucopyranosyl-4-D-glucopyranose							
Formula	C12H22O11							
Mass	342.1162							
Reaction	R00026	R00306	R00952	R01441	R01442	R01443	R01444	R01445
	R02365	R02886						
Enzyme	1.1.99.18	2.4.1.20	2.7.1.85	3.2.1.4				
	3.2.1.21	3.2.1.74	3.2.1.91	5.1.3.11				

Fig. 1. Results obtained by probing KEGG Compound with C00221 and C00185 respectively

³ From KEGG [15]. Portions of these pages have been removed to simplify the discussion and to save space, while remaining true to the challenges encountered.

⁴ In certain cases, probing a source with an imprecise keyword leads to a disambiguation step. This is a separate research issue and we don't address it in this paper. We assume exact keywords are used to perform the search, as explained in Figure 2.

⁵ W3C. Document Object Model. <http://www.w3.org/DOM/>

and uses XPath⁶ for performing the above operations on the DOM tree. The output of the wrapper is a collection of XPath expressions, each pointing to a label and associated data region.

We now explain each step of the algorithm in detail using our running example. Each step is annotated to its corresponding location in the algorithm in Figure 2.

```

Input: n Web pages P
Output: R: L => Xgr    //L is a set of labels. Xgr is a set of XPath's to data entries. R is a map
                        from each label ℓ in L to each data XPath dℓ in Xgr.

Start
For each sample page pi in P {
1   For each text entry t in pi
2       If t is unique to pi,      Add t to Di;
       Else                          Add t to Oi; }
3   For each pi in P {
       Xid = get_XPath(Di);
       Xio = get_XPath(Oi); }
4   Di', Oi', Xio', Xid' = reclassify(Xio, Xid);    // grow the data regions, and reclassify data
   For each XPath dℓ in Xid' {
5       Find closest XPath ℓ̂ in Xio';    // search for XPath of most suitable label in O
       If the corresponding text (label) in Oi' is not in R
           X = {dℓ};    // X is a set containing all data XPath's associated with one label
       Else
           X = X ∪ {dℓ};
       R = ℓ̂ => X;    // XPath of label is mapped to set of corresponding data XPath's
   For each ℓ̂ in R {
6       Generalize corresponding X to Xg;    // Create a single XPath from all paths in set X
7       Find relative path Xgr from ℓ̂ to Xg;    // relative path from label to data
       Replace X with Xgr;
       Replace ℓ̂ with ℓ;    // replace Xpath with the corresponding label
   }
}
End

```

Fig. 2. Wrapper Induction Algorithm

1. The two HTML pages are converted to well-formed XHTML using an HTML-parsing library, i.e. TagSoup⁷, so that standard XML tools can be applied to them. Finally, each page is *screen-scraped* to obtain a set T of values contained in all text nodes. Both sets (T₁ and T₂ in our example) thus contain a union of presentation text, labels and data entries.

2. We compare both sets to initially classify some data entries. Mutually exclusive entries in T₁ and T₂ are classified as data entries (D₁, D₂), and the remaining as non-data entries, or “Other” (O₁, O₂). For example:

$$D_1 = \{C00221, \text{beta-D-Glucose}, \dots, R01520, 1.1.1.47, \dots\}$$

$$D_2 = \{C00185, \text{Cellobiose}, \dots, R00306, 1.1.99.18, \dots\}$$

⁶ W3C. XML Path Language (XPath 2.0) Recommendation. <http://www.w3.org/TR/xpath20/>

⁷ A SAX-compliant HTML Parser. <http://home.ccil.org/~cowan/XML/tag soup/>

$O_1 = \{\text{Entry, Name, ..., Reaction, R00026, Enzyme, ..., 3.2.1.21}\}$
 $O_2 = \{\text{Entry, Name, ..., Reaction, R00026, Enzyme, ..., 3.2.1.21}\}$

Notice that since the data entry R00026 occurs in both instance pages, it is erroneously classified as *Other* at this point.

3. We compute XPath expressions for each entry in the above sets. The expression determines the unique path along the DOM tree for the XHTML file, from the root node to the node containing the entry. For example, the XPath for C00221 is:

`html/body/.../code[1]/table[1]/tr[1]/td[1]/code[1]/text()`

4. We use the XPath expressions to reclassify some data entries which might have been wrongly classified in the previous step (such as R00026, 3.2.1.21). This can be considered as *growing* of a data region, whereby data entries are used to reclassify other entries in their vicinity as data. This reclassification step compares an XPath of a data entry with that of an entry not classified as data using the following two rules:

Rule 1 (Last Element Node Rule): If two XPaths are identical and differ only at the ordering of the last element node, and this last element node in the data XPath precedes the last element node in the non-data XPath, the non-data entry is re-classified as data.

This rule can be explained from an example in Figure 3(a). As shown in the figure, this rule uses data elements to automatically grow data regions towards the right in a table-row. While Figure 3 only shows the example of a table, it is important to emphasize that since this rule is independent of tag names, it works on tags other than those associated to HTML tables, for example, downwards in a list or in a succession of anchor tags. For instance, for the latter case, the XPaths for successive anchor tags (even if they are separated by line breaks) could be:

`html/body/a[1], html/body/a[2]`

It is easy to see from the above example that this rule grows data regions in many types of HTML structures.

Rule 2 (Penultimate Element Node Rule) If the two XPaths are identical and differ only at the ordering of the penultimate element node, and this penultimate element node in the data XPath precedes the penultimate element node in the non-data XPath, the non-data entry is re-classified as a data entry

This rule is similar to Rule 1, except it grows data regions down a table-column as shown in Figure 3(b)

We note that our rules for growing data-regions operate in only two directions. This is based on the observation that labels generally occur above or towards the left of data [33]. Therefore, we restrict our re-classification in these two directions.

After this re-classification step, we have the modified sets:

$D_1' = \{\text{C00221, beta-D-Glucose, ..., R01520, 1.1.1.47, ..., 3.2.1.21}\}$
 $D_2' = \{\text{C00185, Cellobiose, ..., R00306, 1.1.99.18, ..., 3.2.1.21}\}$
 $O_1' = \{\text{Entry, Name, ..., Reaction, R00026, Enzyme, ...}\}$
 $O_2' = \{\text{Entry, Name, ..., Reaction, R00026, Enzyme, ...}\}$

Note that R00026 is not re-classified (incorrectly) because there are no data entries which can grow the data region in its direction. The sets of non-data entries, O_1' and O_2' , now contain both presentation text, as well as labels for our data entries.

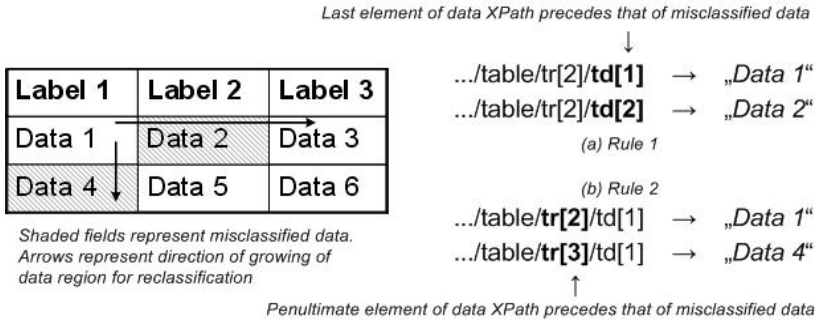


Fig. 3. Growing of data region

```

1 Input: S={C0}, C0=(ℓ01, ℓ02,...); // Set of all page classes discovered. C0 corresponds to the
    initial results page of a deep Web source.
    W = {}; //Set of navigation steps between page classes.
    Output: S'={Ci} (i≥0)
    W={Rij} (i,j≥0, i≠j)
    Start
    S' = S;
    Do{
2   For each C in S' { // for each class in our set
        For each ℓ in C { // for each link-collection associated with a label
            Follow ℓ; // follow the link-collection
3           induceWrapper Cnew; // induce wrapper
4           if (Cnew ∉ S) { // if this is a new class
                Add Cnew in S and S'; // add it to our set
                R = (C → ℓ → Cnew); // form the navigation step
                Add R in W; // add the step
            }
        }
        Remove C from S';
5 } While(S' ≠ NULL) // all classes' link-collections have been explored
    End
    
```

Fig. 4. Site-Wide Wrapper Generation Algorithm

5. For each data entry in a data set, we now select the closest non-data entry as its label. This can be achieved by comparing XPath of the data entry against the XPaths of the non-data entries. The closer a non-data entry is to a data entry, the more element nodes in their corresponding XPath expressions will be matched before a mismatch. The closest element will have the longest common leading path, which is classified as the label. For example, the XPath for data entry 1.1.1.47 is given by:

html/.../table[1]/tr[8]/td[1]/.../code[1]/a[1]

Some XPath expressions for the set of non-data entries include, for example:

html/.../table[1]/tr[6]/th[1]/.../code[1]/ ("Reaction")
 html/.../table[1]/tr[8]/th[1]/.../code[1]/ ("Enzyme")

The latter XPath has the longest sequence of matching nodes with the XPath of our data element (indicated by the bold-face font above). Therefore, the label (“Enzyme”) and corresponding XPath are associated with data entry 1.1.1.47 and its XPath (This association is represented by the “=>” symbol in Figure 4).

Note that multiple data elements can be associated with a single label in such a manner, as shown in Table 1. The last row of the table shows that the data entry R00026, which has been misclassified previously, has been selected as a label for data entries R01520, R01521 etc, as it found to be the closest non-data entry.

Table 1. Two labels inferred, with corresponding data entries and their XPath expressions

Label – With XPath	Data Entry	XPath of Data Entry
Enzyme html/.../th[1]/.../code[1]/	- 1.1.1.47	html/.../td[1]/.../code[1]/a[1]
	1.1.3.4	html/.../td[1]/.../code[1]/a[2]

R00026 html/.../th[1]/.../code[1]/a[1]	- R01520	html/.../td[1]/.../code[1]/a[2]
	R01521	html/.../td[1]/.../code[1]/a[3]

6. The XPaths of data entries classified to the same label are then generalized to form a single XPath expression. The XPaths for entries in Table 1 can be generalized as:

```
html/.../table[1]/tr[8]/td[1]/.../code[1]/a[position()≥1]/text()
html/.../table[1]/tr[6]/td[1]/.../code[1]/a[position()≥2]/text()
```

The last XPath expression above, for example, selects all text entries pointed to by a collection of anchor tags, starting from the second anchor tag. This is required as sample pages may only contain a small number of multiple data entries associated to a label. What is required is that we recognize and generalize that multiple number of data entries are associated for that label, rather than the number of data entries *seen* by the wrapper induction algorithm.

7. A relative path from the label to its corresponding generalized data path is computed. For the “Enzyme” label in Table 1, the relative path to its data is:

```
../.../.../td[1]/.../code[1]/a[position()≥2]/text() (1)
```

Finally, the XPath for the label is replaced with an *anchored* XPath expression, i.e., an XPath which directly access the text node, and does not utilize ancestor nodes. For the “Enzyme” label:

```
//*[text()='Enzyme'] (2)
```

Concatenating (1) with (2) gives us, for label “Enzyme”:

```
//*[text()='Enzyme']../.../td[1]/.../code[1]/a[position()≥2]/text()
```

The wrapper, thus, comprises of a collection of labels associated with a generalized anchored XPath expression to extract corresponding data.

Discussion. It is worth noting here that the wrapper learnt by our algorithm is not tied to the structure of a class of pages. The wrapper anchors, or pivots, to a particular

label, and finds a relative path from the label to associated data entries. As noted in section 2, data pages returned by Web databases can be very dynamic. Our approach is beneficial in this case, as well as in the case that a Web site undergoes a template redesign, for instance. As long as the relative path from a label to its corresponding data remains the same, there would be no need to re-learn the wrapper. The only other limitation is that of labels remaining constant, and as we mentioned in section 2, changes in names of real-world entities, such as biological concepts, is extremely rare. Finally, recall that the wrapper-induction process for our running example results in the misclassification of “R00026” as a label. We discuss a technique to automatically detect and possibly correct such errors in section 3.3.

3.2 Site-Wide Wrapper Induction

As we noted in section 1, data-intensive sites, such as those in the Life Sciences domain, have their data scattered across multiple pages. Therefore, we need a wrapper-induction strategy that extracts data from multiple pages, which might belong to different page classes. This implies that we need to not only discover which pages returned by the server belong to the same class, but also to distinguish between classes and navigational steps between them. We make the following observations:

1. Not all pages of a Web site contain data, for example, pages pointed to by navigational menus, help pages, contact information etc. Therefore, we do not wish to discover all page classes. Rather, we wish to perform *targeted crawling* to only seek out *data-pages* and discover their classes.
2. We observe the concept of *link-collection* [9], which refers to anchor links in a page (class) that share the same path in the DOM tree, from the root element to their parent or grandparent element. As a result, these hyperlinks appear grouped together in the rendered page. In our example of Figure 1, the links on the reaction names form a link-collection, as well as those on the enzyme names. A link collection may be a singleton as well, comprising only a single hyperlink. We also note that hyperlinks in such a collection might not point to the same class of pages. For example, links in a navigation bar or those in categorization menus. However, link-collections that have been *classified* as being over data regions point to the same class of pages. For example, all hyperlinks on enzyme names point to “enzyme details” pages.
3. Pages belonging to the same class contain similar set of labels. However, due to the highly dynamic nature of pages, some labels may be omitted (e.g. NULL values in databases), but their ordering typically remains the same, as they are generated by scripts. If the order of labels on two pages is different, then their page-structures will most likely be different as well. Furthermore, it is highly unlikely that a site will have two different templates to display the same set of labels in the same order.

We base our approach for site-wide wrapper induction on these assumptions:

1. Given the initial result output of a deep Web source, all data-intensive pages can be reached by iteratively following link-collections that occur on data regions. This assumption allows us to do targeted crawling for data-intensive pages, and eliminate navigation bars and menus etc.
2. A pre-classified link-collection points to the same class of pages.

3. Classes of pages can be distinguished from each other based on the labels they contain, and their order.

We now model our site-wide wrapper-induction problem as follows:

A page class C_i is defined by $C_i = \text{SEQ}_i$, where $\text{SEQ}_i = (\ell_{i1}, \ell_{i2}, \dots)$, a sequence of labels $\ell_{i1}, \ell_{i2}, \dots$ described in page class C_i in this order of arrangement. These labels may have link-collections associated with them. Two classes C_i and C_j are considered not equal if $\text{SEQ}_i \neq \text{SEQ}_j$.

Our site model is given by a collection of navigation steps:

$$R_{ijn} = C_i \rightarrow \ell_{im} \rightarrow C_j \quad (i \neq j, i, j, n, m \geq 0)$$

Where ℓ_{im} is the m -th label in page class C_i , the associated link-collection of which points to pages of class C_j .

The goal of the site-wide wrapper induction algorithm is to find the following:

1. $C_i \neq C_j \quad (i \neq j, i, j \geq 0)$
2. $R_{ijn} = C_i \rightarrow \ell_{im} \rightarrow C_j \quad (i \neq j, i, j, n, m \geq 0)$

That is, all possible data-intensive page classes, and the navigation steps between them. The algorithm for site-wide wrapper induction is presented in Figure 5. We explain the algorithm using our example of Figure 1. Each step explained below is annotated to its corresponding location in the algorithm in Figure 4.

1. The algorithm starts with an input of the initial page class C_0 , which corresponds to the initial response pages of the Web source. In our case, this is the page class that is generated by the sample pages of our running example, shown in Figure 1.
2. For each label in this class, corresponding link-collections are followed. Assuming we follow the link-collection of “Enzyme”, a sample result is shown in Figure 5.

Entry	EC 2.7.1.2	Enzyme
Name	glucokinase; glucokinase (phosphorylating)	
Class	Transferases; Transferring phosphorus-containing groups; Phosphotransferases with an alcohol group as acceptor	

Fig. 5. Small excerpts of pages obtained from following “Enzyme” link-collection

3. According to our assumption 2, these pages belong to the same class. We learn a wrapper for this page class using our algorithm in Figure 2, with these sample pages as input. We note that not all links in a link-collection need to be followed. Our initial experiments have shown that ~9 sample pages yield a very good result (section 4).
4. If this wrapper learning process results in a new class, according to our assumption 3, we add this new class to our sets, and define its corresponding navigational steps. In our example, a new class is created, $C_1 = (\text{Entry}, \text{Name}, \text{Class}, \dots)$, as well as a navigation step $R = (C_0 \rightarrow \text{“Enzyme”} \rightarrow C_1)$.
5. The above steps are repeated for each new page class that is learnt in step 3.

3.3 Error-Detection by Mutual Reinforcement

The natural residual output of our site-wide wrapper-induction approach is labeled data. These labels and data can be used to automatically detect and possibly correct

Entry	R00026	Reaction
Name	beta-D-Glucoside glucohydrolase	
Definition	Cellobiose + H ₂ O <=> 2 beta-D-Glucose	
Equation	c00185 + c00001 <=> 2 c00221	

Fig. 6. Top-most portion of “Reaction” page of R00026 From KEGG

errors in our wrapper-induction method for a page-class. We observe that some data entries reappear on different page classes. For example, the enzyme classification numbers in Figures 1 and 6. If the reappearing entries have been correctly classified as data across different page-class wrapper induction runs, then this enforces our confidence that the classification is correct. On the other hand, if, for example, some entries are classified as labels or presentation text by some wrappers and data by others, then this points to a misclassification. This indicates that not enough sample pages were available to distinguish between data, labels and presentation text. We can address this by providing more samples for these page classes. We call such a mismatch as *label-data mismatch*. For example, recall from Section 3.1 that the data entry R00026 was misclassified as a label in our running example (for page class C_0). When we follow the “Reaction” link-collection, we come across the page shown in Figure 6. While learning the wrapper for this class of pages (C_1), R00026 will be (correctly) classified as data. Based on this mismatch, we introduce more learning pages for both C_0 and C_1 . In our example, any page for class C_0 which does not contain “R00026” as an entry will force our algorithm to classify this entry as data, thereby correcting the label to “Reaction” as well. The other type of mismatch is *label-label mismatch*, where the same data entry is assigned different labels across page classes. Recall our observation from section 2 that the same data entries can be labeled differently across different page classes. This can be observed from Figures 1 and 7, where R00026 is labeled as “Reaction” and “Entry”, respectively. Therefore it is impossible to detect whether a label-label mismatch was an error or a correct classification. We slightly modify our site-wide wrapper induction algorithm to incorporate automatic error detection and correction for label-data mismatches. We introduce this mutual reinforcement step each time a new page class is created. The entries classified as data in this new class are compared with labels of previously formed page classes. If a mismatch is found, more sample pages for this new class and conflicting page class are introduced until the mismatch is resolved.

4 Results and Evaluation

We have developed a prototype in Java which implements our algorithms. We use it to perform some preliminary experiments on three real-world biochemical sources, namely KEGG[17], ChEBI[12] and MSDChem[13]. All these sources provide basic qualitative data, and are often used for reference or annotation in more specialized domains. We use a simple random sample of input values for search forms of these

sites in order to probe and induce their initial results page⁸. A Web-crawler based on httpUnit⁹ was manually configured to submit the search forms with these values.

4.1 Page-Level Wrapper Induction Results

We verify the accuracy of our wrappers by applying them to sets of 20 test pages. We manually count the total number of data entries and note corresponding labels across these test pages a priori, and determine the precision and recall of our algorithm in retrieving these data entries and classifying them with the right label.

Before discussing the results, we briefly explain some conditions under which false-negatives and false-positives occur. False-positives generally occur when there is unlabeled data present in the pages. These data usually occur at the top of a page, such as a heading or a large caption, and are often *redundant* data entries, as they reappear as labeled data later in the same page (e.g. compound identification numbers etc). False-positives also occur when data entries are misclassified as labels (as “R00026” in section 3.1). This *may* also result in a corresponding false-negative for the *missed* label (“Enzyme” by misclassification of “R00026”). False-negatives also occur when sample pages do not contain the labels. This is a limitation for all wrapper induction approaches – you can only learn what you see. Our results are summarized in Table 2. The wrapper for the page class belonging to KEGG Reaction has a perfect precision and recall, with a relatively small training set. This owes to the fact that there are frequent pages in KEGG Reaction which contain all labels. Our algorithm is thus able to correctly induce a wrapper for this class. The wrappers for KEGG Compound and ChEBI are unable to retrieve data entries corresponding to the labels which were not learnt (“Sequence” and “IN Number” respectively). Manual inspection of the training set revealed that none of our 15 training pages contained those labels, which leads us to believe that they occur rarely. This can only be removed with more sample pages. The wrapper for MSDChem has false-positives as a result of false classification of redundant data entries to some presentation text in the learning phase. The results for MSDChem are quite interesting, as they demonstrate the usefulness of our *data-region growing* approach. Unlike other sources, pages in MSDChem have a very static structure – no labels are omitted from the pages when corresponding data entry is NULL, as shown in Figure 8. This means that the frequency of data fields being NULL (or “Not Assigned” in this example) is very high. Such fields are not classified as data in our algorithm, as they are constant across many pages. However, we note that the label-data pairs are arranged in a (invisible) table, as shown in Figure 7. Therefore, through rule 2 in Section 3.1, an entry classified as data at the top of the data column in Figure 8 reclassifies all entries below it in the column as data. This accounts for perfect recall, with 3 sample pages.

Overall, we observe that we can get very high precision and recall (~99%, ~98% respectively) from ~9 samples. The precision can be improved with more samples, especially if they contain rarely occurring labels.

⁸ The values can be collected from downloadable flat files or Web services provided by each source.

⁹ A Java library for automated testing of Web sites. <http://httpunit.sourceforge.net/>

Table 2. Results from applying wrappers to 20 test pages each. (L=Labels, T=data entries in 20 test pages, S=samples, R=Retrieved but not classified correctly, IR=Incorrect Retrieval).

SOURCE	#L	#T	#S	TP	FN	FP		P	R
						#R	#IR		
KEGG Compound http://www.genome.jp/kegg/compound/	10	762	3	411	351	46	0	89.9%	53.9%
			6	638	124	39	0	94.2%	83.7%
			9	759	3	0	0	100%	99.6%
			12	759	3	0	0	100%	99.6%
			15	759	3	0	0	100%	99.6%
KEGG Reaction http://www.genome.jp/kegg/reaction/	10	205	3	173	32	0	0	100%	84.4%
			6	205	0	0	0	100%	100%
			9	205	0	0	0	100%	100%
			12	205	0	0	0	100%	100%
			15	205	0	0	0	100%	100%
ChEBI http://www.ebi.ac.uk/chebi	22	831	3	595	236	41	0	93.5%	71.6%
			6	713	118	0	0	100%	85.8%
			9	809	22	0	0	100%	97.3%
			12	829	2	0	0	100%	99.7%
			15	829	2	0	0	100%	99.7%
MSDChem http://www.ebi.ac.uk/msd-srv/msdchem/cgi-bin/cgi.pl	30	600	3	600	0	0	20	96.7%	100%
			6	600	0	0	20	96.7%	100%
			9	600	0	0	20	96.7%	100%
			12	600	0	0	20	96.7%	100%
			15	600	0	0	20	96.7%	100%
Average (based on final wrappers for each source)								99.1%	99.8%

Formula	C10 H16 N5 O13 P3
Defined at	1999-07-08
Last modified at	2007-08-16
EBI name	ADENOSINE-5'-TRIPHOSPHATE
EBI Id	not assigned
Additional name	ADENOSINE-5'-TRIPHOSPHATE
Classification	NUCLEOTIDES
Cas reg number	not assigned
Therapeutic category	not assigned
Merck Id	not assigned
Polymer topology	not assigned
Polymer code	not assigned
Polymer sub type	not assigned
Hetgroup type	NON-POLYMER
Obsoleted	not assigned
Parent	not assigned
Topological variant	not assigned

Fig. 7. Portion of MSDChem page for “ATP”, showing unassigned values

4.2 Site-Wide Wrapper Induction

In this section, we present our results for site-structure discovery, which together with the wrapper induction algorithm constitutes our site-wide wrapper induction approach. We manually model all three sources, which involves manually determining classes for data pages for a source, and the navigation steps for generating these classes. MSDchem and ChEBI have relatively simple models. KEGG on the other hand has a very complex model. It actually consist of a number of back-end database schemas, each having its

unique Web interface, with more than 30 page classes. For this paper, we restrict our manual model to a specific sub-site (KEGG Compound, Drug, Reaction, RPair, Enzyme and Orthology).

Next we use our site-wide wrapper induction algorithm to learn the corresponding wrappers. We restrict our algorithm from exploring the KEGG portal outside our manually defined boundary, allowing it to discover navigation steps within the aforementioned sub-site. Finally, we apply our site-wide wrappers to the three sources to extract data. We limit the execution so that our system extracts data from only 20 test instances of each class of the Web site. We manually count the total number of data entries and note corresponding labels across all test pages a priori, and determine the accuracy of the algorithm, with the results shown in Table 3.

Table 3. Site-wide wrapper evaluation. (#C = Total number of classes, #C' = Number of classes discovered, T = data entries in 20 test pages).

SOURCE	#C	#C'	#T	TP	FN	FP	P	R
MSDChem	1	1	N/A	N/A	N/A	N/A	N/A	N/A
ChEBI	3	1	1711	1195	516	0	100%	69.8%
KEGG	10	7	6223	5044	1179	188	97%	81.1%
Average							98.5%	75.5%

We observe that for MSDChem, even though the navigation steps constituting the site model are correct, the site-wide wrapper induction fails. Upon inspection, we notice that the navigation step from a page instance actually results in the same instance. This implies that the MSDChem Web site consists of a large number of leaf nodes only, having no hyperlinks connecting them to each other. For ChEBI, we have a perfect precision, but a low recall. This indicates that the two classes our algorithm failed to retrieve had rich data regions. Our algorithm also fails to retrieve 3 classes in the KEGG sub-site, though a relatively higher recall suggests these missing classes did not contain as big a data region as in the case of ChEBI. Furthermore, we have some misclassifications in some page wrappers for KEGG which slightly lower the precision for the corresponding site-wide wrapper for KEGG. Overall, a relatively high precision suggests that our assumption that all link-collections associated with data regions point to classes of pages containing data is indeed correct. However, the relatively low recall seems to suggest that we need to relax the restriction that only link-collections associated with data regions should be followed. Instead, links close to data regions should also be followed.

5 Related Work

The earliest approaches to wrapper induction, including [20] and [26] required manually labeled training sets. Due to the large size of the Web and its dynamic nature, supervised techniques do not scale well. Recent attempts have focused on fully automatic wrapper induction techniques. The reader is instructed to read [22] for a survey on wrapper induction techniques. RoadRunner [8, 10] is an automatic wrapper induction algorithm that is closest to our approach, as it uses multiple sample

pages of a page-class. However, unlike our approach, it compares the structures of the sample pages to learn a regular expression, which takes into consideration the mismatches between text and HTML tags across the samples. This regular expression based wrapper is thus tied directly to the page structure. As we noted in section 2, pages from deep Web sources are often very dynamic, where concepts that are NULL are often omitted. RoadRunner would thus require a large number of sample pages, covering all possible types of such omissions, so that its regular expression can accommodate for this dynamic behavior. Lixto [3] and W4F [28] use XPath-like languages “Elog” and “HEL” respectively, and both offer visual tools for creating wrappers in an unsupervised manner. The user selects data of interest in a Web page, and a path from the root of the page to the target node is generated in the respective languages. Therefore, manual identification of data elements is required for each page, which can be laborious for pages containing numerous data entries. ANDES [25] is based on XPath and requires the user to manually provide XPath expressions to extract data. [1] builds on ANDES to induce the XPath expressions using tree traversal patterns but requires annotated examples. IEPAD [6], DeLA [33], ViNTs [27], DEPTA [35] and ViPER [29] are unsupervised wrapper induction techniques that are all based on one common assumption: Data regions in Web pages are constituted by at least two spatially consecutive records that are structurally and visibly similar. This assumption partially holds for result pages of search engines, online listings and E-commerce Web sites, but not for scientific repositories on the Web, as is apparent from our example in Figure 1. Even in the case of E-commerce sites and listings, the initial response pages of a search do exhibit a repetitive structure comprising of records, but the *details* pages describing each result do not exhibit this repetitiveness. All approaches cited above perform wrapper induction on a single class of pages, whereas our approach attempts to automatically classify pages in a Web site into appropriate classes, learn wrappers for each class and discover rules for applying these wrappers on Web pages encountered on the Web site. IDE [36] extracts structured data from different classes of Web pages. It starts with one labeled training page, indicating the information to be extracted. It proceeds to extract corresponding data from test pages based on the similarity between a consecutive sequence of tags before and after the labeled data and the data in the test pages. Whenever extraction fails for a page, it is manually labeled. This requires foreknowledge about which information must be extracted, and assumes that the same information is present and to be extracted from all classes.

We are only aware of one approach to automatic site-structure discovery [9], which also constitutes the main motivation for our approach. The focus of their work is slightly different from ours. It tries to efficiently discover the entire site-structure, whereas we focus on discovering only data-rich regions. Their approach to clustering of Web pages into classes is based on the assumption that pages belonging to the same class contain link-collections that are in a structurally similar arrangement. Based on structural similarity of these link collections, they group Web pages into classes. This is a good assumption for sites that do not have *leaf* pages which do not have any links, such as help pages, FAQs, contacts, legal disclaimers etc. In the absence of hyperlinks, all these pages would be classified into a single class (because their link-collections have the same structure), even though these pages may exhibit considerable structural variations. Our approach is also based on an assumption over

link-collections, but contrary to [9], we assume that link-collections classified to the same label point to pages belonging to the same class. A closely related research field is that of focused crawling, which can either be content-based or structure-based. Content based-crawlers [4] fetch Web pages relevant to a given topic, which is specified by example Web pages. Structure-driven crawling relies on the structural similarity between given sample pages and pages of a Web site to find similar pages [30] or between the pages encountered in a Web site to cluster similar pages [11].

6 Conclusions

We have described a novel wrapper induction technique to extract labeled data from data-intensive Web pages of deep Web sources. The approach takes advantage of the peculiarities typically associated with Life Science Web sites, most notably that they contain labeled data. Our approach is unique in that it automatically classifies structurally similar pages into classes which can then be used for learning wrappers. Navigation steps that are retrieved during the site-wide wrapper induction phase are used to associate wrappers to classes of pages, allowing us to automatically select and apply a wrapper for a page in the Web site. The approach is fully automatic, the samples required for page-level wrapper induction are collected automatically and do not require any manual labeling. The approach does not need fine-tuning of any heuristics or parameters, but does require the presence of labels.

References

1. Anton, T.: XPath-Wrapper Induction by generalizing tree traversal patterns. In: Workshop on Web Mining, in ECML/PKDD (2006)
2. Barbosa, L., Freire, J.: Searching for Hidden-Web Databases. In: WebDB, pp. 1–6 (2005)
3. Baumgartner, R., Flesca, S., Gottlob, G.: Visual Web Information Extraction with Lixto. In: Proc. 27th Internatl. Conference on Very Large Data Bases, pp. 119–128 (2001)
4. Chakrabarti, S., et al.: Mining the Web's link structure. *Computer* 32(8), 60–67 (1999)
5. Chang, K.C.-C., Cho, J.: Accessing the Web: From Search to Integration. In: Proceedings of the 2006 ACM SIGMOD Conference (2006)
6. Chang, C.-H., Hsu, C.-N., Lui, S.-C.: Automatic information extraction from semi-structured web pages by pattern discovery. *SCI expanded* 35(1), 129–147 (2003), Special Issue on Web Retrieval and Mining
7. Chang, K.C.-C., He, B., Zhang, Z.: Mining Semantics for Large Scale Integration on the Web: Evidences, Insights and Challenges. *SIGKDD Explorations* 6(2), 67–76 (2004)
8. Crescenzi, V., Mecca, G., Merialdo, P.: RoadRunner: Towards Automatic Data Extraction from Large Web Sites. In: VLDB, pp. 109–118 (2001)
9. Crescenzi, V., Merialdo, P., Missier, P.: Clustering Web pages based on their structure. *Data & Knowledge Engineering* 54, 279–299 (2005)
10. Crescenzi, V., Mecca, G., Merialdo, P.: Improving the expressiveness of ROADRUNNER. In: SEBD, pp. 62–69 (2004)
11. de Castro Reis, D., et al.: Automatic web news extraction using tree edit distance. In: WWW13, pp. 502–511 (2004)

12. Degtyarenko, K., et al.: ChEBI: a database and ontology for chemical entities of biological interest. *Nucleic Acids Res.* 350, D344–D350 (2008)
13. Golovin, A., et al.: E-MSD: an integrated data. *Nucleic Acids Research* 32(Database issue), 211–216 (2004)
14. He, B., Chang, K.C.-C.: Statistical Schema Matching across Web Query Interfaces. In: *SIGMOD Conference*, pp. 217–228 (2003)
15. He, H., Meng, W., Yu, C.T., Wu, Z.: WISE-Integrator: An Automatic Integrator of Web Search Interfaces for E-Commerce. In: *VLDB*, pp. 357–368 (2003)
16. He, B., Tao, T., Chang, K.C.-C.: Organizing structured web sources by query schemas: a clustering approach. In: *CIKM*, pp. 22–31 (2004)
17. Kanehisa, M.: The KEGG database. In: *Novartis Found Symp.*, vol. 247, pp. 91–101, discussion 101–3, 119–28, 244–52 (2002)
18. Knoblock, C., Kambhampati, C.: Information Integration on the Web. In: *AAAI* (2002)
19. Kabra, G., Li, C., Chang, K.C.C.: Query Routing: Finding Ways in the Maze of the DeepWeb. In: *WIRI 2005*, pp. 64–73 (2005)
20. Kushmerick, N.: Wrapper Induction for information extraction. In: *ICAI* (1998)
21. Kushmerick, N.: Learning to Invoke Web Forms. In: Meersman, R., Tari, Z., Schmidt, D.C. (eds.) *CoopIS 2003, DOA 2003, and ODBASE 2003*. LNCS, vol. 2888, pp. 997–1013. Springer, Heidelberg (2003)
22. Laender, A.H.F., Ribeiro-Neto, B., Silva, A.S.D., Teixeira, J.S.: A brief survey of web data extraction tools. *ACM SIGMOD Record* 31(2), 84–93 (2002)
23. Lu, Y., et al.: Clustering e-commerce search engines based on search interface pages using WISE-Cluster. *Data Knowl. Eng.* 59(2), 231–246 (2006)
24. Madhavan, J., et al.: Corpus-based Schema Matching. In: *ICDE*, pp. 57–68 (2005)
25. Myllymaki, J., Jackson, J.: Robust Web Data Extraction with XML Path Expressions. *IBM Research Report* (2002)
26. Muslea, I., Minton, S., Knoblock, C.: Stalker: Learning extraction rules for semistructured, web-based information sources. In: *AAAI 1998: AI and Information Integration Workshop* (1998)
27. Meng, W., Raghavan, V., Yu, C.: Fully automatic wrapper generation for search engines. In: *WWW14* (2005)
28. Sahuguet, A., Azavant, F.: Building intelligent Web applications using lightweight wrappers. *Data Knowl. Eng.* 36(3), 283–316 (2001)
29. Simon, K., Lausen, G.: ViPER: augmenting automatic information extraction with visual perceptions. In: *CIKM 2005*, pp. 381–388 (2005)
30. Vidal, A., et al.: Structure-driven crawler generation by example. In: *SIGIR 2006*, pp. 292–299 (2006)
31. Wang, J., Wen, J.-R., Lochovsky, F.H., Ma, W.-Y.: Instance-based Schema Matching for Web Databases by Domain-specific Query Probing. In: *VLDB*, pp. 408–419 (2004)
32. Wu, W., Doan, A., Yu, C.T.: WebIQ: Learning from the Web to Match Deep-Web Query Interfaces. In: *ICDE*, p. 44 (2006)
33. Wang, J., Lochovsky, F.H.: Data extraction and label assignment for web databases. In: *WWW12*, p. 187–196 (2003)
34. Zhang, Z., He, B., Chang, K.C.-C.: Understanding Web Query Interfaces: Best-Effort Parsing with Hidden Syntax. In: *SIGMOD Conference*, pp. 107–118 (2004)
35. Zhai, Y., Liu, B.: Automatic Wrapper Generation Using Tree Matching and Partial Tree Alignment. In: *AAAI 2006*, Boston, USA, July 16–20 (2006)
36. Zhai, Y., Liu, B.: Extracting Web Data Using Instance-Based Learning. In: *WWW16* (2007)

An Adaptive Combination of Matchers: Application to the Mapping of Biological Ontologies for Genome Annotation

Bastien Rance¹, Jean-François Gibrat², and Christine Froidevaux¹

¹ LRI, Univ. Paris-Sud, CNRS UMR 8623, F-91405 Orsay, France
bastien.rance@lri.fr, christine.froidevaux@lri.fr

² INRA, Unité Mathématique, Informatique et Génome UR 1077,
F-78352 Jouy-en-Josas, France
jean-francois.gibrat@inra.jouy.fr

Abstract. Biological ontologies are widely used for genome annotation. Identifying correspondences between concepts of two ontologies (mapping) allows the reuse and sharing of annotations. Accordingly, biological ontology mapping has attracted a lot of interest. In this paper, we introduce O'Browser, a semi-automatic method for mapping two functional hierarchies using two sets of carefully annotated proteins. While being based on a classical ontology mapping architecture, O'Browser computes correspondences using a combination of different kinds of matchers. A key feature of O'Browser is that it places the expert at the center of the mapping process at two stages: (i) both to validate the very strong correspondences discovered by the system and to identify functional groups of concepts and (ii) to validate the correspondences given by the combination of results found by the matchers. These matchers have been designed in O'Browser to fit best with functional hierarchy features. For instance, we have introduced a new instance-based matcher which uses homology relationships between proteins. The combination of the different matchers is based on an original notion of adaptive weighting. Here, we show the ability of O'Browser to map concepts of Subtilist to concepts of FunCat, two functional hierarchies. First results appear to be very promising.

Keywords: Ontology mapping, instance-based matcher, functional annotation of genomes.

1 Introduction

With the availability of genomic data for an increasing number of organisms it becomes critical for biologists to be able to compare their functional properties. Given the amount of data involved, this requires systematic computational analyses to be carried out. Therefore the functional annotations must be expressed in a formal and standardized form. Biological concepts need to be organized in structured classifications exhibiting good coverage (in breadth and depth) and computer readability. Hierarchical classifications and ontologies are well-adapted to these tasks. They also provide a controlled vocabulary for the concepts of the field.

The first standardized functional annotation schemes were simple hierarchies in which the specificity of the function description increased when moving from the top of the hierarchy to the leaves [1]. The first classifications were narrowly tied to the need of annotating particular model organisms, such as Multifun for *Escherichia coli* [2] and SubtiList for *Bacillus subtilis* [3]. Therefore, they were relatively biased towards the characteristics of these organisms (in particular, the fact that *E. coli* and *B. subtilis* are bacteria) and dependent on the centers of interest of the biologists that conceived them. Later on, more general hierarchies were designed to take into consideration both prokaryotic and eukaryotic organisms, such as FunCat [4] that was created to annotate genomes at MIPS (Munich Information center for Protein Sequences).

Aside from hierarchical, tree-like, classifications, a different scheme has been proposed known as Gene Ontology (GO) [5]. Originally, GO has been developed for the purpose of comparing the annotation of multi-cellular organisms (*Caenorhabditis elegans* – a worm, *Drosophila melanogaster* – the fruit fly, and others). It consists of three classifications: molecular function, biological process and cellular component. They are structured by hierarchical relations: “is a” and “part of”. GO is represented by independent directed acyclic graphs, although recent attempts to discover associative relations across its three hierarchies have been made [6]. GO, being the most comprehensive classification, has rapidly become a standard. It is by far the most widely used bio-ontology, despite the fact it is relatively unwieldy for prokaryote annotation purposes, having been designed to compare multi-cellular organisms.

Organisms have often been annotated using different classifications. In order to compare meaningfully annotations of organisms that are based on different functional classifications one must first establish correspondences between the concepts of all these classifications (mappings). Indeed, the work we present here was motivated by the need of comparing genomes annotated by INRA groups (using SubtiList) with genomes annotated by MIPS (using FunCat). It is a special case of ontology mapping, since functional classifications are simple ontologies (hierarchical structures with only “is a” relationships), and associations are one-to-one (we intend to link each term of one ontology with a term of another one).

Comparing ontologies for functional annotation is not straightforward, as each ontology reflects expertise of the scientists who have designed it, and, often focuses on their centers of interest. Indeed, some aspects of protein functions are more detailed than others [4]. Some ontologies provide the user with a fine granular description of proteins (e.g. GO), while others are more concise (e.g. FunCat).

In the computer science community, many ontology mapping techniques have been proposed during the last decade. Recently, several surveys have attempted to classify the different techniques [7,8] and a website gathering these techniques has been set up [9]. Since 2006, a series of dedicated international workshops have been organized for tackling the problem of building efficient and accurate mappings between ontologies. Especially, the ISWC 2008 Ontology Mapping workshop has proposed a list of challenges for ontology mapping [10].

In this paper, we adopt the classification of mapping techniques proposed in [7]. Ontology mapping techniques consist in a mixture of elementary matchers and combinations of matchers. Elementary matchers can be distinguished according to two features:

the granularity and interpretation of input on the one hand and the type of input on the other hand. Elementary matchers implement different strategies that can be based on terminology, structure, instances or can use external resources. Generally the matchers compute similarities between the terms of both ontologies, and then combine and filter the results, before submitting the obtained correspondences to the expert of the domain. Examples of matching systems are Anchor-PROMPT [11], COMA++ [12] or FALCON [13]. Regarding the validation of the resulting mappings, it is worth mentioning the Ontology Alignment Evaluation Initiative (OAEI) [14].

We propose a new approach which takes into account the specificity of the ontologies we want to relate. Indeed they have a rather low number of concepts and a very simple structure (hierarchical with only one kind of relationship). Also, the names used to refer to the concepts are often short sentences. Last but not least, concepts of these ontologies do not share common instances (we are not aware of genomes having been annotated with two different functional classifications).

The system we introduce in this paper follows the classical methodology. It is based on a combination of several elementary matchers, each one being defined with the purpose of exploiting a feature of the ontologies studied. Our approach is original insofar as it takes advantage of knowledge of the domain in several aspects and is based on feedback from the user, in a collaborative fashion. For instance, domain experts specify anchors, which are defined as clearly related pairs of concepts. They also define types, which are semantic categories for the concepts within the ontology. The definition of types exploits semantic knowledge about the concepts and meta-knowledge on how the ontologies have been designed, i.e., with which aim, for which organisms, etc. Importantly, experts are asked to validate the results. Another key aspect of our approach lies in proposing a new treatment of instances (since, as mentioned above, these instances are not shared between ontologies) based on evolutionary properties of proteins. Finally, we introduce an original definition of the adaptive weights used for combining matchers.

This paper is organised as follows. Section 2 gives an overview of the O'Browser approach. The main modules of the system are described in Section 3 while results are presented in Section 4. Finally, we compare our work to previous work and draw conclusions.

2 O'Browser Method

For each concept of the source ontology \mathcal{O}_1 , O'Browser looks for the closest concept of the target ontology \mathcal{O}_2 , if any. O'Browser returns a mapping function defined on the set of concepts of \mathcal{O}_1 such that $\mu(C) = (D, \sigma)$ where D is a concept of \mathcal{O}_2 and σ a score which estimates the quality of the correspondence. The function μ is partial as some concepts of the source ontology do not have a corresponding concept in the target ontology. The mapping function is neither injective nor surjective. In addition to the two ontologies, O'Browser takes sets of external biological data (protein domains, GO terms...) as input. Some steps of O'Browser are especially dedicated to functional hierarchies. We will highlight these steps hereafter. By default the other steps deal with directed acyclic graphs.

2.1 A Semi-automatic System

Biologist or bioinformatician experts play a crucial role in O'Browser. They add some biological knowledge to the system on the one hand and validate results on the other hand. These steps are detailed in the next paragraphs.

Anchors: O'Browser's first step is the search for **anchors**. Anchors are pairs of concepts having a strong correspondence to each other based on biological background knowledge and, thus, that must be mapped together. Anchors are identified by using (a) a quasi-identity of names of concepts and (b) homology relationships between the proteins annotated with one of the two concepts. A score based on (a) and (b) is associated to each pair of concepts. Only the best results are considered: a threshold is given by O'Browser's administrator, and only pairs of concepts with a score greater than or equal to this threshold are proposed to the expert. Candidate anchors can be validated or refused by the expert.

A pair can be rejected for three different reasons : (i) the two associated concepts are not comparable at all, as they concern distinct kinds of biological fields (e.g. *Germination* in the case of plants and *Germination* in the case of Prokaryotes), (ii) the two concepts are comparable but the pair of concepts cannot be considered as an anchor because the notions, although close, are nevertheless different (e.g. *Metabolism of DNA* and *Metabolism of RNA*), and (iii) one of the two concepts is too general w.r.t. the second. After this validation step all pairs accepted by the experts are considered as anchors.

Types: Also experts are asked to associate **types** to the concepts of both ontologies whether they are anchors or not. Concepts that are related to the same functional genomic field are assigned to the same type. As previously mentioned, different points of view on protein functions are merged by functional hierarchies (e.g. metabolism, cell localisation...). Concepts of distinct types will never be mapped. Using types to generate candidate pairs of concepts for mapping allows to reduce the search space as not all the pairs will be generated but only pairs of concepts of the same type.

The expert may use the most general concepts of the ontologies to manually identify and name the basic types. Let T be the set of terms denoting these basic types. Types are then propagated automatically to descendant concepts. The system allows concepts to have multiple types. More formally, let \preceq be the subsumption relation between concepts of the ontologies and let O_1 (resp. O_2) be the set of concepts of \mathcal{O}_1 (resp. \mathcal{O}_2).

- $BASE_TYPE \subseteq (O_1 \cup O_2) \times T$
- $TYPE$ is the smallest relation included in $(O_1 \cup O_2) \times T$ that contains $BASE_TYPE$ and is closed for the subsumption relation of O_1 and O_2 :
 - $BASE_TYPE \subseteq TYPE$
 - $\forall C, C' \in O_1 \cup O_2, \forall \tau \in T$
 $[(C' \preceq C \wedge (C, \tau) \in TYPE) \Rightarrow (C', \tau) \in TYPE]$

An example of type chosen by the expert is "Cell process".

From the analysis of the anchors validation step, O'Browser can automatically deduce constraints on types. For instance, if a concept C cannot be mapped to another concept D because C is too specific, then O'Browser deduces that no children of C will ever be mapped to D . These constraints are used in the mapping candidates generation step.

2.2 Matchers

O'Browser uses a set of elementary matchers to evaluate the similarity of two concepts of two hierarchies. Each of these matchers focuses on a kind of similarity between properties of concepts. For instance, names of concepts may be used, or proteins annotated with the concepts or Pfam domains associated to proteins annotated with the concepts.

A matcher defined on a domain Dom takes two concepts including their properties as an input and returns a score as an output.

$$M : O_1 \times O_2 \rightarrow Dom$$

In O'Browser, we used three different kinds of matchers (the first two are instance-based matchers): (1) homology relationships-based matcher, (2) external resources-based matcher and (3) syntactic matcher. These matchers use various techniques and exploit complementary properties of concepts of the ontologies.

(1) Homology Matcher: Evolution is *the* central concept in biology. A consequence of evolution is that all organisms exhibit various degrees of kinship in the tree of life. Here, we are interested in the concept of homology. Two (genes or) proteins in present day organisms are said to be homologous if they descend from a common (gene or) protein ancestor. When two proteins share this evolutionary relationship they have properties in common. The property that interests us here is the fact that they may have kept the function of their ancestor or may have evolved *related* functions.

The *homology* matcher we have developed uses this property. Unlike the instance-based matcher proposed in [15] for the life sciences ontologies, instances (here proteins) of the concepts are not shared. Let $\text{ext}(C)$ be the set of proteins from organisms that have been annotated with concept C of O_1 and let $\text{ext}(D)$ be the set of proteins of other organisms that have been annotated with concept D of O_2 . We have determined the homology between proteins of sets $\text{ext}(C)$ and $\text{ext}(D)$ using BLAST [16], a program that is classically employed for this purpose in biological studies. Let $\text{homologous}(e, f)$ denote the homology relationship between proteins $e \in \text{ext}(C)$ and $f \in \text{ext}(D)$.

Our expectation is the following. If the concepts C and D were totally equivalent we would expect all proteins annotated with C in the first organism to be *exclusively* homologous with proteins annotated with D in the second organism. In practice, we measure the degree of *relatedness* of concepts C and D for the homology as follows:

$$SIM(C, D) = \frac{|\text{ext}(C)|}{|\text{ext}(C)| + |\text{ext}(D)|} * \frac{CDn_{concept}}{Cn_{relation}} + \frac{|\text{ext}(D)|}{|\text{ext}(C)| + |\text{ext}(D)|} * \frac{DCn_{concept}}{Dn_{relation}}$$

where:

- The number of proteins annotated with C involved in homology relationships with a protein annotated with at least one concept of O_2 is:

$$Cn_{relation} = |\{e \in \text{ext}(C), \exists F \in O_2, \exists f \in \text{ext}(F), \text{homologous}(e, f)\}|$$

- The number of proteins of C involved in homology relationships with proteins annotated with D is:

$$CDn_{concept} = |\{e \in \text{ext}(C), \exists d \in \text{ext}(D), \text{homologous}(e, d)\}|$$

Definition of $Dn_{relation}$ and $DCn_{concept}$ are analogous.

(2) **Shared Instances-Based Matcher:** The next category of matcher uses properties shared by the two hierarchies. In reference [15], the authors presented an approach based on this type of data by using the *Kappa* similarity. The similarity we use is much simpler and efficient enough for our data. Biological properties come from various external sources such as Pfam domains or GO terms or SwissProt keywords, etc. The similarity between concepts is evaluated by using a *Hamming* distance between the set of terms associated to the first concept $C \in O_1$ and the set of terms associated to the second concept $D \in O_2$.

Let $\text{prop}(i)$ denote the set of terms associated to a given instance i of a concept for a given biological property (e.g. Pfam domains). We define p_1 and p_2 as follows :

$$p_1 = \{\text{prop}(e), e \in \text{ext}(C)\}$$

$$p_2 = \{\text{prop}(d), d \in \text{ext}(D)\}$$

$$\text{Sim_Hamming}_{\text{prop}}(C, D) = 1 - \frac{|p_1 \cup p_2 - p_1 \cap p_2|}{|p_1 \cup p_2|}$$

(3) **Syntactic Matcher:** Syntactic and string comparison matchers are very useful. They usually work well, though they do not allow to discover all the correspondences. We use two matchers designed to capture two kinds of syntactic similarity. In both cases a pre-treatment is needed to remove the empty-words (non informative words such as “of”, “from”, ...). The first matcher uses a *Levenstein* distance [17] (edit distance between two strings). The second takes into account frequency of each word in the two ontologies. For example, the word “Metabolism” is widely used in both ontologies. It may have a very important impact on the comparison of “Metabolism of Carbon” and “Metabolism of Sulfur”, whereas the two terms are not semantically equivalent. Therefore, each word is weighted by its frequency in both ontologies to express its specificity.

Given two not empty strings s_1 and s_2 , the similarity based on Levenstein distance is evaluated by:

$$1 - \frac{\text{lev}(s_1, s_2)}{\max\{\text{len}(s_1), \text{len}(s_2)\}}$$

where $\text{lev}(s_1, s_2)$ denotes the Levenstein distance between s_1 and s_2 , and $\text{len}(s_1)$ (resp. $\text{len}(s_2)$) the length of s_1 (resp. s_2), and $\text{len}(s_1) + \text{len}(s_2) > 0$.

2.3 Adaptive Weighting

The next stage consists in combining the results of the different matchers. Classical approaches use user-defined factors to weight each matcher. But it may very well happen that the good results of a matcher are spoiled by results of another one. For example, let us assume that the homology matcher provides a good score for a pair of concepts, but that the names of these concepts are not syntactically close. Combining the two

matchers could be worse than using only the homology matcher. In order to overcome this problem, we introduce **adaptive weighting**. We claim that the confidence given in a matcher should partially depend on its results. For instance, knowing that two concepts have a very close name is very informative. Thus, a weight of 1 will be given to good results of the syntactic matcher. On the other hand, the fact that two names of concepts are not syntactically close does not imply that the two concepts are remote. Therefore, the weight of the syntactic matcher will be reduced when it provides average or weak results. That weak weight allows us to give a relative stronger weight to good results provided by another matcher. The weight associated to scores of a pair of concepts given by a matcher is adaptive insofar as it depends on the value of this score. The definition of the weighting function is based on our knowledge on the matcher for the application domain.

We define for each matcher M_i a weighting function W_{M_i} which associates a weight to each score of the matcher. Each weighting function W_{M_i} is designed according to some background knowledge about the matcher M_i . For $M_i : O_1 \times O_2 \rightarrow Dom_i$, we have $W_{M_i} : Dom_i \rightarrow [0, 1]$. For example, if $M_{Homology}(C, D) = 1$, that is, the two concepts C and D are very close, then the role played by the homology matcher is very relevant and its weight for this pair of concepts is 1, while the weight is set to 0.15 for a pair of concepts which has a score less than 0.5.

2.4 Structure-Based Matcher

The last matcher is based on structure. A well-known idea is that the more ancestors and descendants of two concepts in two different ontologies are themselves related, the closer the two concepts. This matcher has been designed in order to exploit the specific tree structure of the functional hierarchies. In a first top-down pass, the influence of ancestors is propagated to their descendants. The second pass is bottom-up, and propagates the influence of descendants. Formally:

Let us define $\pi_0, \pi_1 \in [0, 1]$ s.t. $\pi_0 + \pi_1 = 1$, let $\text{sim}_0(C, D)$ be the weighted sum of scores of matchers for a given pair of concept C and D using the adaptive weighting, or 1 if the pair of concepts is an anchor. We introduce $\text{anc}(C)$ (resp. $\text{anc}(D)$) the set of all the concepts ancestors of C (resp. D), except C (resp. D) itself. The first pass is evaluated using:

$$\begin{aligned} \text{sim}_1(C, D) &= \pi_0 \text{sim}_0(C, D) + \pi_1 \text{sim}_1(\text{anc}(C), \text{anc}(D)) \\ \text{sim}_1(\text{anc}(C), \text{anc}(D)) &= \frac{\sum_{A \in \text{anc}(C)} \sum_{B \in \text{anc}(D)} \omega^{k_1 + k_2} \text{sim}_1(A, B)}{|\text{anc}(C)| |\text{anc}(D)|} \end{aligned}$$

with $k_1 = |\text{depth}(C) - \text{depth}(A)|$, $k_2 = |\text{depth}(D) - \text{depth}(B)|$ and ω a weight $\in [0, 1]$ (the more two ancestors are distant from C and D , the less their role is important). The second pass is evaluated analogously with sim_1 and sim_2 replacing respectively sim_0 and sim_1 , and the set of ancestors replacing the set of descendants.

3 O'Browser Modules

O'Browser takes as input two ontologies and sets of external biological data stored within databases, and gives as output a mapping between concepts of the two

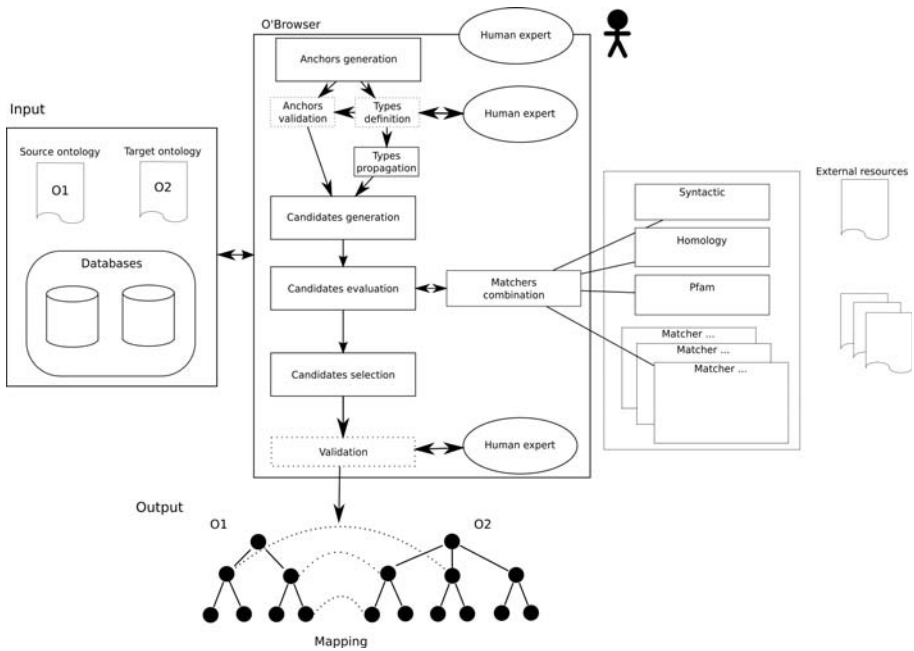


Fig. 1. Architecture of O'Browser

ontologies. The global architecture of O'Browser is shown in Figure 1. It consists of five main modules: the anchors generation previously described and the five modules we present now.

The system is fully implemented in Java 1.6. Further information and screenshots about O'Browser application are available online at the O'Browser website [18].

Anchors validation: We have designed a graphical user interface to help the user during the validation steps, including anchors validation, types definition, and results validation. Figure 2 represents a screenshot of the anchor validation GUI: the two hierarchies are displayed at the bottom of the figure. Candidate anchors are highlighted in the hierarchy trees. Users can access to the proteins annotated with any concept of the hierarchies by a simple click on the corresponding node. As an example, P56398 is annotated with the term “Funcat:12.01.01 ribosomal proteins” as shown on the right-hand side of Figure 2. Finally, when the expert has made his/her own opinion about the candidate anchor, he/she can choose to reject or accept the proposed anchor using the form, see top of Figure 2. Motivation of this choice can be detailed in the comment field.

Candidate generation: Using types defined by the expert and constraints calculated at the anchors validation step, O'Browser builds a set of pairs of concepts, which are likely to be mapped (candidates).

Candidate evaluation: The global score of each candidate pair obtained at the previous step is evaluated using all the elementary matchers scores and the adaptive weighting

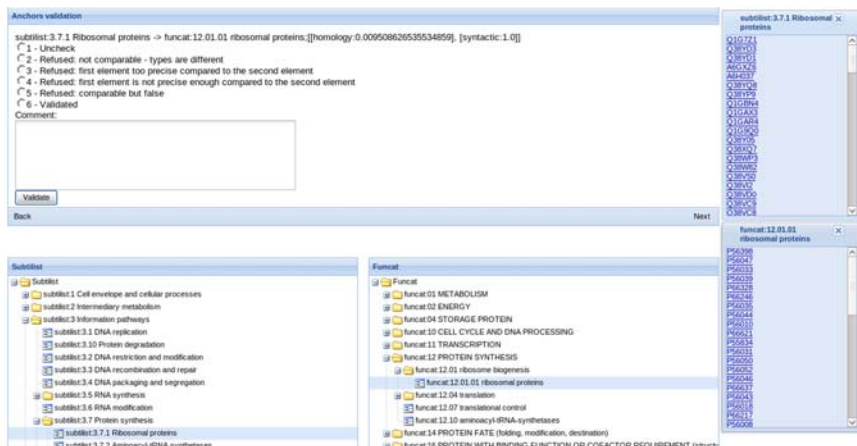


Fig. 2. O'Browser anchors validation GUI

(see section 2.3). This global score is refined by the structural matcher that takes into account influence of ancestors and descendants (section 2.4). For each concept of the source ontology, a list of concepts from the target ontology, ordered by their final score of similarity, is returned to the next module. As this step may take some times (several minutes) O'Browser uses a cache system. Scores of matchers for candidate pairs are stored in O'Browser.

Candidate selection: For each concept C of O_1 , let \mathcal{M}_C denote the set of all scored candidate pairs for C (this set can be empty). The selection module selects the unique pair (C, D) that has the best score in \mathcal{M}_C (if any). O'Browser returns the set of all selected scored pairs.

Candidate validation: The last O'Browser module is the validation module. If a “gold standard” (mapping between concepts built by an expert) is available, obtained results are compared to this standard. If no gold standard is available, a fraction of the results is proposed to the expert for validation. Unlike the anchors validation, a selected mapping can be validated or refused for two different reasons only: the mapping is wrong or one concept is too specific/general w.r.t. the second one. The fraction of results proposed to the expert depends on the size of the data. If the number of concepts of the source ontology is reasonable, all the unvalidated candidate correspondences are proposed to the expert. On the contrary, if the number of concepts is too large, O'Browser randomly chooses concepts which will be validated. Concepts are chosen in various places of the ontologies.

4 Application: Mapping of Sublist to FunCat

4.1 Experimental Settings

We carried out a first test of our system for aligning two functional hierarchies of interest for our collaborators: Sublist and FunCat (version 2.1). A few characteristics of

Table 1. Figures of the functional hierarchies Subtilist and FunCat

Hierarchy	# concepts	Max. depth	# concepts at level 1	# leaves	Max. # children of a concept
Subtilist	63	3	6	54	10
FunCat 2.1	1305	6	28	980	28

the structure of the two hierarchies is given on Table 1. We have used biological data publicly available in association with the two hierarchies, that is, 3 genomes manually annotated using Subtilist and 4 using FunCat. Data associated with proteins from SwissPFam (Pfam version 22.0), GO terms and SwissProt keywords from Uniprot (version 14.5) have also been used. All these data have been stored in the Microbiogenomics data warehouse [19]. Finally, we have used a partial gold standard, manually built by one expert (41 of the 63 concepts of Subtilist are mapped to concepts of FunCat).

All experiments have been performed on an Intel Pentium 1,73 GHz processor with 1 GB of RAM.

4.2 Results

Anchors: O’Browser discovered 28 pairs of concepts likely to be anchors (8 using homology relationships, 24 using syntactic comparison, and 4 using both similarity measures). The expert turned down 5 pairs: in two cases the concepts were not comparable, in two cases one of the concepts was too precise w.r.t. the second, and in the last case the concepts were comparable but not close (concept “subtilist:1.1 Cell Wall” child of “subtilist:1 Cell envelope and cellular processes” and concept “funecat:42.01 Cell Wall” child of concept “funecat:42 BIOGENESIS OF CELLULAR COMPONENTS” are clearly comparable concepts, but the pair cannot be considered as an anchor). After this expert anchor validation step, O’Browser was supplied with 23 anchors. Twenty one out of the 23 anchors were located in the same subtree of Subtilist rooted by “subtilist:3 Information pathways”.

Types: During the anchors validation step, the expert also identified 6 basic types (see Table 2 for further information). Moreover, a few concepts of O_1 and O_2 have been excluded from the mapping process because (i) they did not refer to a precise biological function (e.g. funecat:99 UNCLASSIFIED PROTEINS) or (ii) they had types present in only one of the two ontologies.

Using types and exclusion constraints reduced the number of candidates pairs from 82 215 (the number of concepts of Subtilist times the number of concepts of FunCat) to 12 000.

Matcher settings: We used the matchers previously described. The homology matcher used the BLAST program with the following parameters: proteins with at least 60 % of identical residues after alignment and an E-value of at most 10^{-4} were considered to be homologous. It is worth noticing that we did not have to evaluate the BLAST score of all pairs of proteins on-the-fly, since these results were already available in the Microbiogenomics database.

Table 2. Types description over functional hierarchies

Types	# concepts of Subtilist	# concepts of FunCat
Cell process	17	267
Cell rescue and defense	3	45
Information pathways	21	97
Metabolism	14	310
Transport / binding	8	115
Transposable element	2	7
Excluded from mapping	4	3
Not present in the other hierarchy	0	560

Obtained mapping: Using O’Browser with these settings, we obtained the following results:

- For 80% of the concepts of Subtilist covered by our partial gold standard, the concept selected by O’Browser as the corresponding concept in FunCat is the same as the one proposed by the gold standard.
- For 90% of the concepts covered by our partial gold standard, the corresponding concept given by the gold standard is present among the first top-5 scored concepts obtained by O’Browser at the end of the candidate evaluation step.

A comparison of the results drawn from the various combination strategies is shown in Table 3.

Analysis of O’Browser limitations. Let us analyse thoroughly two cases where O’Browser did not find the best result according to the gold standard.

- *An ancestor or a descendant of the closest concept is found by O’Browser.* Consider the concept “Subtilist:2.1 Metabolism of Carbon”. The expert has mapped the Subtilist concept to the FunCat concept “FunCat:01.05 C-compound and carbohydrates

Table 3. Comparison of results of various combination strategies over the gold standard (41 correspondences)

Strategy	# correspondences found as 1st results of O’Browser	# correspondences found in the top 5 results of O’Browser
Anchors	23	23
Best matcher alone	28	33
Matchers combination without weighting	27	35
Matchers combination with classical weighting	29	35
Matchers combination with adaptive weighting	33	38

metabolism”. The candidate evaluation module has found the FunCat concept at the third place in the ordered list, whereas it found one at the first place of its descendants (Funcat:01.05.01), and at the second place one of its ancestors (Funcat:01). This error was due to the fact that the final scores for these three correspondences were very close.

- *A single correspondence is provided by O’Browser while the obvious expected correspondence is clearly not a one-to-one mapping.* Consider the concept “subtilist:1.2.1 Transport/binding of proteins/peptides”. This concept clearly corresponds to the concepts “funcat:20.01.09 peptide transport”, “funcat:20.01.10 protein transport”, “funcat:16.01 protein binding” and “funcat:16.02 peptide binding”. In the gold standard the expert had to make a choice, but the actual mapping is a one-to-many mapping and should rather be a union of these four concepts. Although this result is unsatisfactory, O’Browser is not really wrong in that case, since it has explicitly been designed to build a one-to-one mapping.

The lack of biological data may be another source of errors for O’Browser (the final score strongly depends on the fact that a sufficient amount of biological data is available).

5 Discussion and Conclusion

In this section, O’Browser is compared to other systems for aligning ontologies. Many works have addressed the ontology matching problem. In the following, we only compare O’Browser with three well-known systems related to it: (1) Anchor-PROMPT; (2) FALCON; (3) COMA++, as far as they may use anchors, partitioning (a notion close to types) and matcher combination. Let us first recall some specificities of functional hierarchies. They have tree structures with only “is-a” relationships between concepts. In addition, the structures of functional hierarchies may differ in several aspects (breadth, depth, and granularity of the trees).

- Anchor-PROMPT [11] is an extension of PROMPT [20]. It uses anchors found with syntactic-based matchers, and extends the search for mapping to concepts close to the anchors. This step is widely used employing various techniques as in FALCON [13]. O’Browser uses an anchor discovering step, but this step is completed with a validation step (whose results are used to automatically find constraints on types). Moreover, we have enriched this step by using not only a syntactic-based matcher but also an homology-based matcher.
- FALCON [13] uses a structure-based partitioning technique. It splits the ontology into small clusters and groups them using anchors, in order to reduce the mapping problem on the initial ontologies to a mapping problem on smaller parts. O’Browser does not exploit the structure to split ontologies in smaller parts: structural information is too poor (only is-a relationships) to allow an interesting partitioning. In our approach, the expert defines basic types using a semantic criterion. The expert-defined types are spread to descendants using the functional hierarchies properties.

- COMA++ [12] is an extension of COMA [21]. The system is based on a combination of different kinds of matchers. COMA++ offers a choice of several strategies to combine matchers (max, average, weighted, min). O'Browser is based on the same architecture, but uses adaptive weighting, a new strategy to combine matchers that exploits the scores of the correspondences obtained for a given matcher.

In this paper we have described O'Browser a new semi-automatic system for finding correspondences between concepts of two functional hierarchies. Information about O'Browser is available online [18]. While being based on a classical ontology mapping architecture, O'Browser computes correspondences using a combination of different kinds of matchers. The expert is placed at the center of the mapping process, first to validate the anchors and then to determine types. We have designed a new matcher based on homology relationships, and introduced adaptive weighting.

We see several opportunities for future work. In particular, we plan to use O'Browser on other biological ontologies, either functional hierarchies or simple bio-ontologies.

As a further prospect, the ability of comparing different functional ontologies will allow us to readily use data from genomes annotated with these ontologies. The availability of such data is crucial for any method whose goal is to automate the annotation process itself [22].

Acknowledgement

We are grateful to Sarah Cohen-Boulakia for her helpful comments on previous version of the paper.

This work was funded partially by the Agence Nationale de la Recherche (ANR ARA Microbiogenomics). B.R. is a PhD student supported by the French Ministry of Research.

References

1. Rison, S., Hodgman, T., Thornton, J.: Comparison of functional annotation schemes for genomes. *Functional & Integrative Genomics* 1, 56–69 (2000)
2. Riley, M.: Systems for categorizing functions of gene products. *Curr. Opin. Struct. Biol.*, 388–392 (1998)
3. Moszer, I., Jones, L., Moreira, S., Fabry, C., Danchin, A.: Subtilist: the reference database for the *Bacillus subtilis* genome. *Nucleic Acids Res.* 30, 62–65 (2002)
4. Ruepp, A., Zollner, A., Maier, D., Albermann, K., Hani, J., Mokrejs, M., Tetko, I., Güldener, U., Mannhaupt, G., Münsterkötter, M., Mewes, H.: The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Res.* 14(32(18)), 5539–5545 (2004)
5. The Gene Ontology Consortium: Creating the gene ontology resource: design and implementation. *Genome Res.* 11, 1425–1433 (2001), <http://www.geneontology.org>
6. Bodenreider, O., Aubry, M., Burgun, A.: Non-lexical approaches to identifying associative relations in the gene ontology. In: *Pacific Symposium on Biocomputing*, pp. 104–115 (2005)
7. Euzenat, J., Shvaiko, P.: *Ontology Matching*. Springer, Heidelberg (2007)
8. Kalfoglou, Y., Schorlemmer, M.: Ontology mapping: The state of the art. In: *Dagstuhl Seminar on Semantic Interoperability and Integration*, vol. 04391 (2005)

9. Ontology Matching web site, <http://www.ontologymatching.org>
10. ISWC 2008 Third International Workshop on Ontology Matching, Karlsruhe, October 26 (2008)
11. Noy, N.F., Musen, M.A.: Anchor-PROMPT: Using non-local context for semantic matching. In: Proceedings of the workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI), pp. 63–70 (2001)
12. Aumüller, D., Do, H.H., Massmann, S., Rahm, E.: Schema and ontology matching with COMA++. In: Proceedings of SIGMOD 2005 (2005)
13. Hu, W., Yuzhong, Q., Gong, C.: Matching large ontologies: A divide-and-conquer approach. *Data Knowl. Eng.* 67(1), 140–160 (2008)
14. Ontology Alignment Evaluation Initiative, <http://www.oaei.ontologymatching.org>
15. Kirsten, T., Thor, A., Rahm, E.: Instance-based matching of large life science ontologies. In: Cohen-Boulakia, S., Tannen, V. (eds.) DILS 2007. LNCS (LNBI), vol. 4544, pp. 172–187. Springer, Heidelberg (2007)
16. Altschul, S., Gish, W., Miller, W., Myers, E., Lipman, D.: Basic local alignment search tool. *J. Mol. Biol.* 215, 403–410 (1990)
17. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* 10, 707–710 (1965)
18. O'Browser website, <http://www.lri.fr/%7erance/obrowser/>
19. Lemoine, F., Labedan, B., Froidevaux, C.: GenoQuery: a new querying module for functional annotation in a genomic warehouse. *Bioinformatics* 24, 322–329 (2008)
20. Noy, N., Musen, M.: PROMPT: Algorithm and tool for automated ontology merging and alignment. In: Proc. of AAAI 2000, pp. 450–455. MIT Press, Cambridge (2000)
21. Do, H.H., Rahm, E.: COMA - a system for flexible combination of schema matching approaches. In: VLDB, pp. 610–621 (2002)
22. Azé, J., Gentils, L., Toffano-Nioche, C., Loux, V., Gibrat, J.F., Bessières, P., Rouveirol, C., Poupon, A., Froidevaux, C.: Towards a semi-automatic functional annotation tool based on decision tree techniques. In: BMC Proceedings, International Workshop on Machine Learning in Systems Biology, MLSB 2007, vol. 2(4) (2007)

Slicing through the Scientific Literature

Christopher J.O. Baker¹, Patrick Lambrix², Jonas Laurila Bergman²,
Rajaraman Kanagasabai³, and Wee Tiong Ang³

¹ Department of Computer Science & Applied Statistics, University of New Brunswick, Canada

² Department of Computer and Information Science, Linköpings universitet, Sweden

³ Data Mining Department, Institute for Infocomm Research,
Agency for Science Technology and Research, Singapore

Abstract. Success in the life sciences depends on access to information in knowledge bases and literature. Finding and extracting the relevant information depends on a user's domain knowledge and the knowledge of the search technology. In this paper we present a system that helps users formulate queries and search the scientific literature. The system coordinates ontologies, knowledge representation, text mining and NLP techniques to generate relevant queries in response to keyword input from the user. Queries are presented in natural language, translated to formal query syntax and issued to a knowledge base of scientific literature, documents or aligned document segments. We describe the components of the system and exemplify using real-world examples.

1 Introduction

Indispensable components of knowledge discovery infrastructures are online repositories of freely available unstructured text from the scientific literature. Information retrieval techniques are commonplace for the harvesting of documents while conversion of document formats to make them amenable to text mining is an ongoing irritation. Text mining techniques are swiftly being deployed in industrial strength platforms albeit with the need for domain specific customization. Yet despite the improving proficiency of text mining tools, text extracts are not always readily accessible to end users without augmentation with semantic metadata.

At the same time existing search paradigms using keyword search over indexes of text summaries continue to limit end users. Users do not know how to use tools that allow them to formulate more expressive queries e.g. queries involving known relations between entities. This amounts to a lack of knowledge of available search technology. Nor do users know the limits of the domain coverage of a given resource that they are querying, for example does PubMed include documents on wearable electronic devices for personal health monitoring or is it beyond its scope. It would surely save users time if they know the extent of the answers they can obtain from a given body of knowledge. This amounts to a lack of knowledge of the domain.

In addition to the challenges posed by lack of semantic annotation to mined raw text fragments and poor cognitive support for query composition, system developers are faced with a lack of reusable domain specific metadata to facilitate semantic annotation. Semantic annotation of text segments relies on the existence of curated domain-specific

controlled vocabularies and the mapping of semantic types in ontologies to canonical named entities. Up until recently the use of sophisticated 'domain' metadata was not widely adopted for the indexing of text segments derived from scientific documents, in part due to the dearth of suitably designed ontologies. Using domain ontologies scripted in W3C standard ontology languages, rich in expressive power and inference capability, to annotate mined text segments makes possible an advanced range of literature navigation and search capabilities.

In addition to the search of documents based on named entities or predicates using keywords, relationships or class names as entry points, metadata specific graph mining can further augment search tools. Moreover customized search applications can be readily developed for a multitude of end user requirements including content recommendation. In this paper we describe infrastructure for navigating literature that provides to users, through a natural language interface, (i) ontology driven query, (ii) context of query terms, (iii) cross domain query which obviates the need for users to have knowledge about query languages and underlying data sources, while providing an overview of the scientific domain, connections between entities in the domain and a comprehensive understanding of decisions in query strategy. Search results are linked to scientific documents, text segments and ontology terms. Our contributions in this paper include the definition of the theoretical foundations for this paradigm as well as a framework for systems based on this paradigm. Further, we illustrate this paradigm with some examples.

The remainder of the paper is organized as follows. In section 2 we describe an example scenario for how our new search paradigm can be used. The theoretical foundations and a framework for systems following the paradigm are developed in sections 3 and 4, respectively. Further, we show an implementation of the framework in section 5 and use this implementation to revisit our example scenario (section 6). Related work is given in section 7.

2 Example Scenario

A user performs a keyword search, for example, 'lipid'. In current systems all documents containing the word 'lipid' are retrieved. Some systems that implement ontology-based querying, may also retrieve the documents that contain words representing sub-concepts of lipid. The user, however, is not interested solely in retrieving these documents or abstracts, but also wants to investigate relationships between lipids and other concepts. The problem is that she does not know what relevant questions can be asked. Consulting ontologies for properties can provide this knowledge. For instance, in Lipid Ontology 'lipid' is related to a number of other concepts, such as to 'protein' via the relation 'interacts with' and to diseases based on the relation 'implicated in'. The user would want the system to describe the query 'which proteins interact with lipids that are implicated in a disease?' and make this query available, along with other relevant lipid related queries, to the user in the form of a natural language query. In addition, the user would want to be able to access the information and context relevant to one or more keywords. This context would include connections between the keywords (possibly via other terms) as well as terms that are related to the keywords. Again,

ontologies can provide for a context by investigating the neighborhoods and connections of the keywords to other terms.

We may even need multiple ontologies, for instance, to find the answer to the query 'Which lipids interact with proteins (from the lipid ontology) that are involved in signal processing (proteins involved in signal processing - from the signal ontology) and are implicated in causing a disease (implicated_in from the lipid ontology).

Moreover the ontologies can be instantiated with named entities extracted from scientific documents. This can generate the response that certain oxidized polyunsaturated fatty acids are involved in phosphorylating p53, which is involved in apoptosis, and implicated in ovarian cancer which is derived from one or more text segments instantiated to one or more ontologies. Furthermore because of the addition of expressive literature metadata to the ontologies, querying for provenance information (documents / journals / authors in which the information was contained) is made possible. We could ask natural language queries like 'In which documents have lipids interacting with signaling proteins, and known to cause disease, been found ?'

3 Theoretical Foundations

One of the main foundations of our paradigm is the use of ontologies. Intuitively, ontologies (e.g. [12]) can be seen as defining the basic terms and relations of a domain of interest, as well as the rules for combining these terms and relations. In our approach ontologies are used to guide the user in asking relevant questions. The intuition is that the ontologies define the domain knowledge including our knowledge about the connections between different terms in the domain. Therefore, the relevant queries regarding a set of terms are the ones for which the terms are connected in the ontology. In this section we formalize the notion of 'relevant queries'.

3.1 Slices

Many ontologies in the life sciences can be represented by graphs where the concepts are represented as nodes and the relations (including is-a relations) are represented as edges. In this case, given an ontology, a relevant query including a number of concepts and relations from the ontology can be seen as a connected sub-graph of the ontology that includes the nodes representing the given concepts and the edges representing the given relations. We define this formally using the notion of *query graph*. Further, the set of query graphs including a number of concepts and relations from the ontology is called a *slice*.

¹ If the terms are not connected, then they are not useful for guiding the user in asking relevant questions based on the ontology. However, in that case a sub-set of the query terms may still be relevant. Also, if a co-occurrence of terms (even without relations) is useful for the user, then traditional search approaches can be used.

² We assume undirected edges. An edge represents a relation and its inverse.

³ For instantiated ontologies these definitions can be extended to also handle instances, by allowing nodes to represent instances. In our implementation (see section 5) we do allow queries to the knowledge base (see section 4) that involve instances.

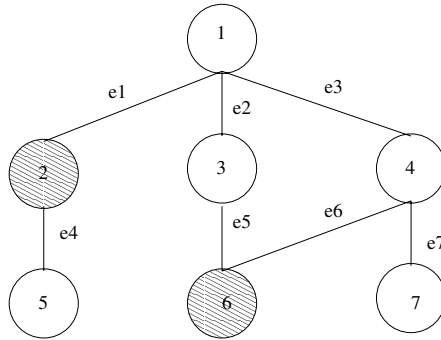


Fig. 1. Ontologies and query graphs

Definition 1. Given a graph $G = (N, E)$ where N is the set of nodes in G and E the set of edges, and a set of nodes $C \subset N$ and a set of edges $R \subset E$, a **query graph in G based on C and R** is defined as a connected sub-graph $G' = (N', E')$ of G where $C \subset N'$ and $R \subset E'$.

Definition 2. Given a graph $G = (N, E)$ where N is the set of nodes in G and E the set of edges, and a set of nodes $C \subset N$ and a set of vertices $R \subset E$, a **slice in G based on C and R** is defined as the set of query graphs in G based on C and R .

As an example, consider the ontology graph in figure 1 and assume query terms related to nodes 2 and 6. Then there are several relevant queries i.e. query graphs in this ontology based on nodes 2 and 6. For instance, the sub-graph containing nodes 2, 1, 3 and 6 and edges e1, e2 and e5 is a query graph based on nodes 2 and 6. Another query graph consists of nodes 2, 1, 4 and 6 and edges e1, e3 and e6. A slice based on nodes 2 and 6 is then the set of all possible query graphs based on nodes 2 and 6.

There are a number of special cases of this definition. (i) When two or more concepts, but no relations are given ($R = \emptyset$), in this case we are looking for relevant queries containing given concepts only. While most keyword search algorithms would try to find documents in which multiple terms co-occur, in this case there is an extra requirement that there are connections, albeit un-specified, between the search terms in the ontology, thereby augmenting the relevance of the returned documents; (ii) Where only a single concept and no relations are provided, in this case a slice represents all the relevant queries in which the query term features. Instead of just returning all documents containing the query term, this approach allows a user to browse the ontological environment of the term. It also guides the user in asking more specific questions, thereby removing many of the irrelevant documents.

3.2 Aligned Ontology Slices

In cases where we want to retrieve information that covers different but related domains or when we want to integrate information from different views on one domain, one

ontology does not suffice. Queries will comprise of terms from different overlapping ontologies. Therefore an alignment i.e. a set of mappings between terms of overlapping ontologies, must be available. In the biomedical domain, for instance the Bioportal (bioportal.bioontology.org, [15]) repository of ontologies stores mappings between different ontologies and this can be used. If an alignment between the used ontologies is unavailable ontology alignment systems (e.g. overviews in [11,17,14,9], the ontology matching book [5], and the ontology matching web site at <http://www.ontologymatching.org/>) may be used for finding mappings.

When an alignment between the ontologies is given, we can deal with a query including terms from overlapping ontologies by connecting the part of a query using the terms in one ontology to the part of the query using the terms in another ontology through a mapping in the alignment.

In the definitions below, we first define an alignment in terms of a graph and then define query graphs with parts in two different ontologies.

Definition 3. An **alignment** between $G1 = (N1, E1)$ and $G2 = (N2, E2)$ is a set of mappings between nodes in $G1$ and nodes in $G2$. The alignment is represented by a graph (NA, EA) such that

- (i) $NA \subset N1 \cup N2$,
- (ii) each edge in EA connects a node in $N1 \cap NA$ with a node in $N2 \cap NA$,
- and (iii) each node in NA is connected to another node in NA through an edge in EA .

The definition states that an alignment (set of mappings) is represented by a graph such that (i) the alignment graph uses only nodes from the source ontologies, (ii) an edge in the alignment graph represents a mapping between a node in the first ontology and a node in the second ontology, and (iii) every node in the alignment graph should participate in a mapping.

Definition 4. Let $G1 = (N1, E1)$ and $G1Q = (NQ1, EQ1)$ be a query graph in $G1$ based on $C1$ and $R1$. Let $G2 = (N2, E2)$ and $G2Q = (NQ2, EQ2)$ be a query graph in $G2$ based on $C2$ and $R2$. Let $A = (NA, EA)$ be an alignment between $G1$ and $G2$. An **aligned query graph based on $G1Q$ and $G2Q$ given A** is a connected graph $G = (N, E)$ such that

- (i) $N \subset N1 \cup N2$, $E \subset E1 \cup E2 \cup EA$,
- (ii) $NQ1 \subset N$, $EQ1 \subset E$, $NQ2 \subset N$, $EQ2 \subset E$,
- and (iii) $\exists n_1 \in NQ1$, $n_2 \in NQ2$, $n_{1a} \in N1 \cap NA \cap N$, $n_{2a} \in N2 \cap NA \cap N$, $e_a \in EA$ such that: there is a path in $G \cap G1$ from n_1 to n_{1a} and a path from n_{2a} to n_2 in $G \cap G2$, and e_a is an edge between n_{1a} and n_{2a} in $G \cap A$.

The definition states that: (i) the nodes in the aligned query graph belong to the source ontologies, and the edges in the aligned query graph belong to the source ontologies or to the alignment, (ii) the nodes and edges in the original query graphs are included in the aligned query graph, and (iii) the original query graphs are connected by at least one path going through a mapping in the alignment.

As an example, consider the ontology graphs and alignment in figure 2. The alignment between the two ontologies is given by the mappings 4-E and 7-F. The query terms are represented by nodes 2, 6, A and D. The sub-graph containing nodes 2, 1, 3, and 6 and edges e11, e12 and e15 is a query graph based on nodes 2 and 6 in the first ontology.

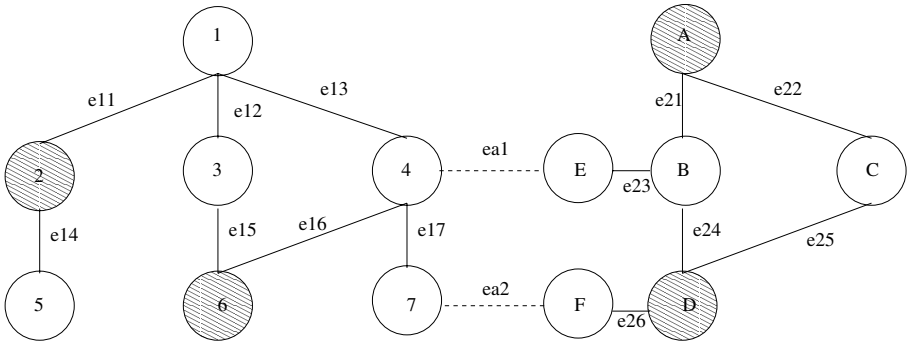


Fig. 2. Aligned ontologies and aligned query graphs

The sub-graph containing nodes A, B and D and edges e21 and e24 is a query graph based on nodes A and D in the second ontology. We can connect these two query graphs via the path containing the nodes 6, 4, E and B and the edges e16, ea1 and e23. One part of this graph is included in the first ontology, another part in the second ontology and a third part in the alignment. Therefore, one possible aligned query graph includes the nodes 2, 1, 3, 6, 4, E, B, A and D and the edges e11, e12, e15, e16, ea1, e23, e21 and e24. (Another possible aligned query graph may make use of the mapping 7-D.)

An aligned slice represents a set of aligned query graphs.

Definition 5. Let $S1$ be a slice in $G1 = (N1, E1)$ based on $C1$ and $R1$ and $S2$ be a slice in $G2 = (N2, E2)$ based on $C2$ and $R2$. Let $A = (NA, EA)$ be an alignment between $G1$ and $G2$. An **aligned slice for $S1$ and $S2$ given A** is defined as the set of aligned query graphs based on the query graphs in $S1$ and $S2$ given A .

4 Framework

Using the theoretical foundations in section 3 we now proceed to define a framework for systems supporting our new search paradigm for literature document bases (see figure 3). As input the user gives a number of query terms. A first output is a list of suggestions for queries in natural language that are relevant with respect to the query terms and the ontologies. These queries can then be run and results are returned. A result can be in the form of knowledge extracted from the documents in the literature base or as documents or document segments.

External Resources. The first external resource is the *literature document base*. It contains the documents that can be searched. The second external resource is an *ontology and ontology alignment repository*. It contains ontologies in which the query terms can be found as well as established alignments between the ontologies.

Computed Resources. The *knowledge base* contains instantiated ontologies. The instantiation represents two kinds of information. First, knowledge, in the form of named entities and relations from the literature, is extracted and normalised to canonical names

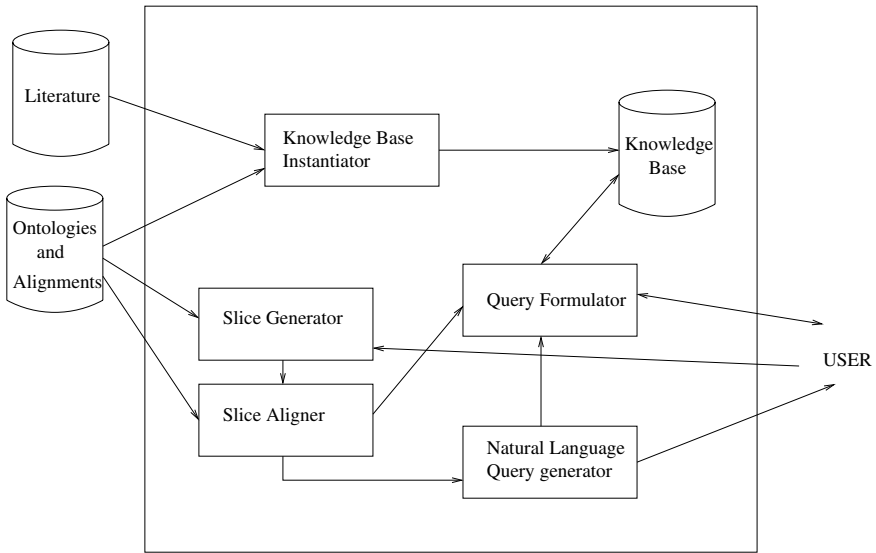


Fig. 3. Framework

- from which concept instances and relation instances are generated. This a form of semantic annotation. Further, the ontology instances are connected to instances of provenance documents and document segments in which they occur.

Process - Knowledge base Instantiation. The *knowledge base instantiator* creates the knowledge base with knowledge extracted from the literature and connections of the ontology terms to documents or document segments. This component relies on different sub-components such as entity recognizers and text mining modules.

Process - Slice Generation and Alignment. The user drives the generation of slices by providing one or more query terms. For each of the ontologies in the ontology repository, the *slice generator* computes, given the query terms included in the ontology, a slice representing the relevant queries that are possible using the provided query terms and the ontology. The resulting slices are given to the *slice aligner*. This component generates aligned slices, representing queries involving terms from different ontologies, using the slices and the alignments in the repository. (If the query terms only occur in one ontology, no alignment is needed and this components just returns the slice.)

Translation - Slice to Query. The (aligned) slices represent sets of queries. The *natural language query generator* translates these queries to queries in human understandable natural language text. This component may have different sub-components such as generation, aggregation and surface realization / grammar checking of the query. These natural language queries are built from the labels of edges and nodes in the slices and presented to the user. Slices are also translated to the equivalent formal query, in syntax of the query language suitable for querying the knowledge base.

Query. When the user chooses a natural language query to be run, the *query formulator* issues the preformulated query representing the slice and the natural language query to the knowledge base and results are returned to the user.

5 Implementation

In this section we describe our current implementation of the framework and use the scenario from section 2 to exemplify the instantiation of the different components.

Literature document base. The literature document base used in our scenario was generated from a collection of 7498 PubMed abstracts that was identified by manual curation to be relevant to the subject of Ovarian Cancer (OC). Within this collection we found 683 papers that had lipid names from which 241 full papers were downloadable. Retrieved research papers were converted from their original formats to ascii text.

Ontologies. The ontologies that we used for this scenario are lipid ontology [2] and our version of the signal ontology. An alignment between these ontologies was generated using the ontology alignment system SAMBO [11].

Knowledge base. As representation language for the knowledge base we used OWL and adopted the conceptualization developed in our previous work [2] in which a Literature Specification of document metadata was introduced to the Lipid Conceptualization making it possible to instantiate simple axioms such as Lipid *Occurs_in* Sentence. The knowledge base instances are generated from full texts provided by the content acquisition engine using the BioText toolkit (<http://datam.i2r.a-star.edu.sg/~kanagasa/BioText/>).

The instantiation of the knowledge base comprises of three stages: concept instance generation, property instance generation, and population of instances. Concept instances are generated by first extracting the name entities from the texts and then normalizing to canonical names and grounding them to the ontology concepts. We used a gazetteer that processes documents and recognizes entities by matching term dictionaries against the tokens of processed text, tagging the terms found [10].

We used the lipid name dictionary described in [2] which was a custom synthesis of terms from a Lipid Data Warehouse that contains lipid names from LIPIDMAPS, Lipid-Bank and KEGG, IUPAC names, and optionally broad synonyms and exact synonyms. The manually curated Protein name list from Swiss-Prot (<http://au.expasy.org/prot/>) was used for the protein name dictionary. A disease name list was created from the Disease Ontology of the Centre for Genetic Medicine (<http://diseaseontology.sourceforge.net>).

To evaluate the performance of our named entity/concept recognition we constructed a gold standard corpus of 10 full-texts papers related to the apoptosis (which is central to understanding ovarian cancer). We extracted 119 sentences and tagged the mentions of Protein name and Disease name. In these sentences we annotated all valid mentions of the two concepts and built the corpus. To evaluate performance of named entity/concept recognition a corpus without the concept annotations was passed to our text mining engine and the concepts recognized. Our system was evaluated in terms of precision and recall. Precision was defined as the fraction of correct concepts recognized over the total number of concepts output, and recall was defined as the fraction of concepts recognized among all correct concepts. The evaluation of entity recognition, in Table 1, shows that our text mining achieved performance comparable to that of the state-of-the-art dictionary-based approaches. In our future work, we plan to make use of advanced entity recognition techniques, e.g. fuzzy term matching and co-reference resolution, and also train our system on larger corpora, to address these issues.

Table 1. Precision and recall of named entity recognition

Named Entities	Mentions		Precision	Recall
	Target	Returned		
Disease	32	37	0.54	0.62
Lipid	58	25	0.96	0.47
Protein	269	181	0.76	0.51
Micro average			0.75	0.51

Our normalization and grounding strategy is as follows. Protein names were normalized to the canonical names entry in Swiss-Prot. Object property and Datatype property instances are generated separately. From the Lipid, Protein and Disease instances, four types of relation pairs namely Lipid-Protein, Lipid-Disease, Protein-Protein, and Protein-Disease are extracted. For relation detection, we adopt a constraint-based association mining approach whereby two entities are said to be related if they co-occur in a sentence and satisfy a set of specified rules. This approach is detailed in [2].

The concept instances are instantiated to the respective ontology classes (as tagged by the gazetteer), the Object Property instances to the respective Object Properties and the Datatype property instances to the respective Datatype properties. This was automated using a custom script developed with the OWL programming framework, JENA API (<http://jena.sourceforge.net/>) for this purpose.

Slices. Given a number of query terms matching ontology terms from one ontology, the query graphs (slices) based on these terms are generated. For efficiency reasons our algorithm generates multiple query graphs at the same time, thereby computing slices immediately. Slices can be represented by graphs as well: a slice can be represented by $G_s=(N_s, E_s)$ where N_s is the set of all nodes in all query graphs in the slice and E_s is the set of all edges in all query graphs in the slice.

We have currently focused on slices based on concepts, i.e. all query terms represent concepts and not relations. Our algorithm, which is an extension of the algorithm proposed in [1], starts from the given concepts and traverses the ontology graph in a depth-first manner to find paths between the given concepts. These paths can be put together to find the slices.

Slice alignment. Our implemented algorithm computes an important sub-set of the aligned slice as defined in definition 5. As input we use two slices represented as graphs, the original ontologies as well as an alignment. The algorithm generates the shortest paths from the concepts in the first ontology on which the first slice is based, to concepts in the second ontology on which the second slice is based, via concepts in the alignment. This heuristic implements the intuition that the shorter paths represent closer relationships between the concepts than longer paths. For instance, in figure 2, possible shortest paths would be 6 - e16 - 4 - ea1 - E - e23 - B - e21 - A, 6 - e16 - 4 - ea1 - E - e23 - B - e24 - D, and 6 - e16 - 4 - e17 - 7 - ea2 - F - e26 - D. The original slices together with these shortest paths constitute our result.

Natural language query generation. The aligned slice is then translated into natural language (see [1] for details). For each query graph contained in the aligned slice we generate a natural language query for consumption by domain experts. The input to the natural language query generation (NLQG) sub-system are aligned slices which

are represented by as set of triples. Each triple represents an edge and its end-nodes in the aligned slice: $\langle \text{ARG1}, \text{PREDICATE}, \text{ARG2} \rangle$ represents the concepts ARG1 and ARG2 that are related to each other by the relation PREDICATE. To translate a triple into natural language, we primarily use a template-based NLQG methodology. In this approach, the domain-specific knowledge and language-specific knowledge required for NLG are encoded as rule templates. The rule templates were generated using a rule learning algorithm. Given a triple, a content determination module recognizes the domain entities in the triples and extracts them for use as content terms. Upper level entities such as concepts and relations are identified and extracted directly via a rule template. For extracting the lower level entities, e.g. the verb and noun in an object property, we employ the BioText toolkit to perform part of speech tagging and term extraction. This is done as a preprocessing of the triples and the results are passed to a rule matching engine. The rule matching engine applies the best matching rule and retrieves a corresponding template to generate the natural language query. When two or more text components are generated they are aggregated to generate a compact query. We employ a set of aggregation patterns that are applied recursively to combine two or more queries sharing the same entity as a conjunction, as well as a generalized aggregation pattern that employs property hierarchy for combination. In the final step the query statement is checked after after sentence aggregation for grammar and generates a human understandable query. We employ an open source grammar checker, called LanguageTool (<http://www.languagetool.org/>), which is part of the OpenOffice suite. We added several rules to enrich the grammar verification checker.

Query result. After (aligned) slices are generated, in addition to being translated into natural language for consumption by end users, their graph triples are formulated into the corresponding syntax of the A-box query language (nRQL) of the reasoning engine RACER [7]. (For details we refer to [11].) nRQL is an A-box query language for the description logic $\mathcal{ALCQHI}_{\mathcal{R}^+}(\mathcal{D}^-)$. All queries written in natural language have a corresponding syntactic version that is issued to the knowledge base. A range of queries can be formulated based on the slices generated by user input. Complex queries are formulated based on multiple triples found in a graph and their connection is based on whether each set of domain and range in different predicates has similar properties. At least one triple and an optional set of domain and range in a role assertion query are necessary. In addition the specification of joins between multiple triples, representing conjunction of predicates, unknowns (variables) and constraints is necessary. For instance, the query graph in figure 4 represents the natural language query 'Which proteins interact with lipids that are implicated in a disease?'. The nRQL format of the query is

```
(RETRIEVE (?X ?Y ?Z)
  (AND (?X Protein) (?Y Lipid) (?Z Disease)
    (?X ?Y Interacts_with)
    (?Y ?Z Implicated_in)))
```

Our implemented system also allows queries to the knowledge base that involve instances. For instance, if we replace variable ?Z in the query above with the instance Ovarian Cancer, then this constrains the retrieval of all the instances of Protein to those that interact with a Lipid instance that is implicated in Ovarian Cancer.

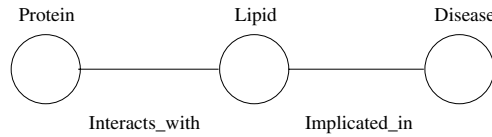


Fig. 4. Query graph example

6 Example Scenario Revisited

Given the Lipid ontology and our version of signal ontology, and the literature base as described in section 5, our system has instantiated a knowledge base. Upon a keyword query by the user for 'lipid', aligned slices are generated involving the lipid concept. The aligned query graphs in the slices are translated to natural language as well as to formal queries. The system then presents relevant queries involving lipid to the user. Examples of such queries are shown in figure 5. The user may learn about the ontological environment of lipid through the generated queries.

The user may be interested in the query 'Which lipid is implicated in a disease and interacts with proteins involved in signal pathways?' Running this query will result in a nRQL query to the knowledge base and an answer is returned: the lipid unsaturated fatty

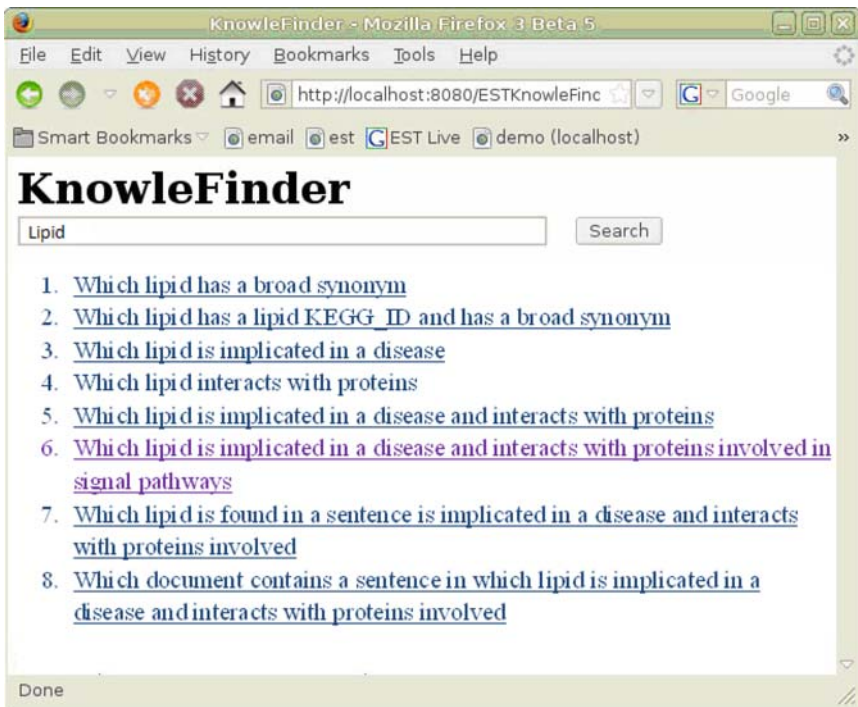


Fig. 5. Generated queries

Question

NLG: Which lipid is implicated in a disease and interacts with proteins involved in signal pathways ?

```
nRQL: (RETRIEVE (?X ?Y ?Z ?W)
(AND (?X Protein) (?Y Lipid) (?Z Disease) (?W SignalPathway)
(?X ?Y Interacts_with) (?Y ?Z Implicated_in) (?X ?W Involved_in)))
```

Result

Protein	Lipid	Disease	Signal Pathway
P53	Unsat. Fatty Acid	Ovarian Cancer	Apoptosis

Fig. 6. Query and answer

acids interacts with the protein p53, which is involved in apoptosis, and is implicated in Ovarian Cancer (see figure 6).

7 Related Work

We are not aware of any other work that fully deals with the problems of lack of knowledge of the domain and lack of knowledge of search technology. There are a number of systems that do tackle parts of these problems.

There exist very few systems that allow natural language querying. An example is askMEDLINE [6] that allows a user to pose a query in natural language and essentially uses the MeSH terms and other eligible terms in the natural language query to query PubMed using PubMed Entrez' E-Utilities. Although this helps the user with the lack of knowledge of available search technology, it does not alleviate the problem of lack of knowledge of the domain.

To aid the user in query formulation, [2] designed and deployed an interactive graphical query tool, Knowlegator, for the construction of queries using axioms provided in the ontology. Manipulation of these 'query atoms' from the OWL-DL ontology invokes A-box queries to a reasoner and subsequent ontology interrogation. The benefits of this paradigm include the ease of use and extensibility of query complexity far beyond the typical keyword searches and to a degree of query complexity suitable for domain experts who ask complex questions but who have limited agility with query syntax of various query languages. [10] extended this approach by taking advantage of transitive properties in populated ontologies to rebuild apoptosis pathways using protein entities mined from texts about apoptosis. A graph mining algorithm with graphical support for, (i) the selection of two pathway endpoints and (ii) rendering of pathways, was developed on top of the existing query tool, Knowlegator. It was also further customized to support bulk queries and rendering for all lipid-protein interactions relevant to a chosen pathway.

There are a number of systems that use ontologies to organize search results and allow a user to browse the literature via the ontology terms. For instance, GoPubMed [4]

uses ontologies to index PubMed abstracts. Upon a keyword query, for each ontology term the number of PubMed abstracts containing the term or one of its descendants is computed. The results can then be browsed using the ontology. These systems alleviate the lack of knowledge of the domain problem, as the user can browse the results based on co-occurrence of the query term with ontology terms. However, it is still up to the user to decide whether this co-occurrence is relevant or accidental. Also, these systems usually do not deal with multiple ontologies and their overlap.

There are a number of systems that use text mining and extract knowledge from documents based on ontologies. For instance, upon a keyword query, EBIMed [16] retrieves abstracts from Medline and finds sentences that contain biomedical terminology in the result. The terminology comes from public resources. The sentences and terminology are used to create overview tables representing associations between the different terms. Textpresso [13] splits literature documents into sentences and words and labels them using ontology terms. The allowed queries are and/or combinations of keywords and labels.

Natural language generation technology is a mature technology dating back 10 years and is now being deployed in commercial settings, such as for providing query options to electronic health records [8]. Recently there have been initiatives aiming to produce textual summaries from Semantic Web ontologies. In the main they address how existing NLG tools can be adapted to take Semantic Web ontologies as their input. In their chapter [3] Bontcheva and Davis describe limitations of three such systems and highlight that quality of the generated text is highly dependent on the ontological constructs in the ontology and how their semantics is interpreted and rendered by the NLG system. Moreover before addressing knowledge transfer and NLG issues a re-assessment of appropriate metrics for evaluation may be required.

8 Conclusion

In this paper we have tackled the problems of lack of knowledge of available search technology and lack of knowledge of domain that users experience when they search for literature relevant to their task. We have proposed a framework that supports a search paradigm that uses (multiple) ontologies to generate relevant queries based on some keywords, translates these into natural language and allows a user via these natural language queries to query an instantiated knowledge base generated from the literature and the ontologies. We have defined the technical foundations and have described an implementation of the framework and its use.

There are still a number of issues that need further investigation. As our implemented algorithms do not compute the full slices or aligned slices, but use heuristics (e.g. shortest path for aligned slices), we want to investigate the influence of these as well as other heuristics. There is a trade-off between completeness (generating all possible queries) and information overload (showing all possible queries may not be instructive or may even be confusing for the user). Another interesting issue is whether it is possible to define a useful relevance measure for the generated queries, which could be used to rank the queries before showing them to the user. Further, as there is a connection between a slice generated from a set of keywords and a slice generated by a sub-set of this set of

keywords, this connection could be used to optimize the process or to suggest the user possible interesting generalizations or specializations of the topic.

References

1. Ang, W.T., Kanagasabai, R., Baker, C.J.O.: Knowledge translation: Computing the query potential of bioontologies. In: International Workshop on Semantic Web Applications and Tools for Life Sciences (2008)
2. Baker, C.J.O., Kanagasabai, R., Ang, W.T., Veeramani, A., Low, H.S., Wenk, M.R.: Towards ontology-driven navigation of the lipid bibliosphere. *BMC Bioinformatics* 9(suppl. 1), S5 (2008)
3. Bontcheva, K., Davis, B.: Natural language generation from ontologies. In: Davis, Grobelnik, Mladenic (eds.) *Semantic Knowledge Management, Integrating Ontology Management, Knowledge Discovery, and Human Language Technologies*, pp. 113–127. Springer, Heidelberg (2008)
4. Doms, A., Schroeder, M.: GoPubMed - exploring PubMed with the gene ontology. *Nucleic Acids Research* 33(web server issue), W783–W786 (2005)
5. Euzenat, J., Shvaiko, P.: *Ontology Matching*. Springer, Heidelberg (2007)
6. Fontelo, P., Liu, F., Ackerman, M.: askMEDLINE: a free-text, natural language query tool for MEDLINE/PubMed. *BMC Medical Informatics and Decision Making* 5(1) (2005)
7. Haarslev, V., Möller, R., Wessel, M.: Querying the semantic web with Racer + nRQL. In: *Proceedings of the International Workshop on Applications of Description Logics* (2004)
8. Harris, M.D.: Building a large-scale commercial nlg system for an EMR. In: *Proceedings of the Fifth International Natural Language Generation Conference* (2008)
9. Kalfoglou, Y., Schorlemmer, M.: Ontology mapping: the state of the art. *The Knowledge Engineering Review* 18(1), 1–31 (2003)
10. Kanagasabai, R., Low, H.-S., Ang, W.T., Wenk, M.R., Baker, C.J.O.: Ontology-centric navigation of pathway information mined from text. In: *Knowledge in Biology - 11th Annual Bio-Ontologies Meeting collocated with Intelligent Systems for Molecular Biology*, pp. 1–4 (2008)
11. Lambrix, P., Tan, H.: SAMBO - a system for aligning and merging biomedical ontologies. *Journal of Web Semantics* 4(3), 196–206 (2006)
12. Lambrix, P., Tan, H., Jakonienė, V., Strömbäck, L.: Biological ontologies. In: Baker, Cheung (eds.) *Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences*, pp. 85–99. Springer, Heidelberg (2007)
13. Müller, H.-M., Kenny, E.E., Sternberg, P.W.: Textpresso: An ontology-based information retrieval and extraction system for biological literature. *PLoS Biology* 2(11), e309 (2004)
14. Noy, N.F.: Semantic integration: A survey of ontology-based approaches. *Sigmod Record* 33(4), 65–70 (2004)
15. Noy, N.F., Griffith, N., Musen, M.: Collecting community-based mappings in an ontology repository. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) *ISWC 2008. LNCS*, vol. 5318, pp. 371–386. Springer, Heidelberg (2008)
16. Rebholz-Schuhmann, D., Kirsch, H., Arregui, M., Gaudan, S., Riethoven, M., Stoehr, P.: EBIMed - text crunching to gather facts for proteins from Medline. *Bioinformatics* 23, e237–e244 (2007)
17. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches. In: Spaccapietra, S. (ed.) *Journal on Data Semantics IV. LNCS*, vol. 3730, pp. 146–171. Springer, Heidelberg (2005)

Exploiting Parallelism to Accelerate Keyword Search on Deep-Web Sources

Tantan Liu, Fan Wang, and Gagan Agrawal

Department of Computer Science and Engineering
Ohio State University, Columbus OH 43210
{liut,wangfa,agrawal}@cse.ohio-state.edu

Abstract. Increasingly, biological data is being shared over the deep web. Many biological queries can only be answered by successively searching a number of distinct web-sites. This paper introduces a system that exploits parallelization for accelerating search over multiple deep web data sources. An interactive, two-stage multi-threading system is developed to achieve task parallelization, thread parallelization, and pipelined parallelization. We show the effectiveness of our system by considering a number of queries involving SNP datasets. We show that most of the queries can be accelerated significantly by exploiting these three forms of parallelism.

1 Introduction

Biologists today spend large amount of time and effort in querying multiple remote or local data sources. Integration has become an important phase in biology research process, as it allows biologists to combine knowledge from multiple disciplines, or in some case, search multiple data sources within a discipline.

One of the popular online medium for disseminating biological data is the *deep web*. The deep web refers to data sources with backend databases that are only accessible through the query forms. There are more than 1000 online databases in the biological domain and the number is still increasing rapidly every year [1]. One particular challenge in accessing biological information that researchers are interested is that it often requires a search across multiple distinct and inter-dependent deep web databases.

Because deep web queries are executed over a wide area network, they can be quite time consuming. A recent study from a deep web integration system shows that nearly 80% of the execution time is spent on data delivery between the server and the clients [2]. As a biological researcher may need to interact with several data sources for a single query, this cost can be a significant distraction.

In this paper, we describe an approach for accelerating search over deep web data sources. Our approach involves a system design which facilitates *task parallelism*, *thread parallelism*, and *pipelined parallelism*. These refer to searching independent data sources in parallel, using multiple threads to query a single data source, and pipelining partial results to overlapping queries on inter-dependent data sources, respectively.

Our work has been in context of a biological integration system that targets SNP related data [3,2]. Our system design involves a *querier pool* and a *data pool*. We have one querier corresponding to each data source. By carefully coordinating the interactions between queriers for inter-dependent sources, we are able to exploit task, thread, and pipelined parallelism.

We have evaluated our system using 30 queries over 28 data sources. Our results show that the benefit from parallelization varies significantly. The maximum speedup we obtain is 12, and 6 of the queries achieve a speedup of 6 or better. Certain queries have a higher potential for task parallelism. These queries typically involve a large number of data sources and/or attributes, and thus, are more likely to have data sources without inter-dependence. Some other queries can benefit more from thread and pipelined parallelism. This is because of the larger number of input queries to each data source.

2 Background and Problem Definition

The work presented in this paper is in the context of a biological data integration system, which provides keyword search functionality over *deep web* data sources.

The use of deep web for data dissemination is growing rapidly in recent years, especially in the scientific communities. The deep web refers to data sources with backend databases that are only accessible through the query forms. A user needs to fill in the query forms in order to request data from these deep web data sources. When a user submits a query by filling a query form, data related with the query in the backend database are returned in a HTML page dynamically. A number of efforts have been attempting to build deep web systems that can integrate both the query interface and the query results (structured data) of the deep web-sites within a specific domain [3,4,5,6,7].

Biological data is increasingly being shared through deep web data sources. For example, there are more than 1000 online databases in the biological domain and the number is still increasing rapidly every year [1]. One particular challenge in accessing biological information that researchers are interested in is that it often requires a search across multiple different deep web databases. No single database can provide all user requested information, and the output of

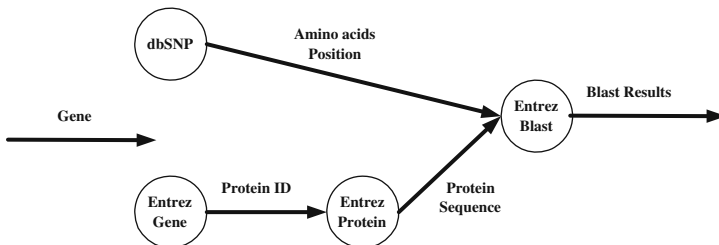


Fig. 1. Graph Representation of a Query Plan

some databases need to be the input for querying another database. Specifically, consider a query that asks for the amino acids occurring at the corresponding position in the orthologous gene of non-human mammals with respect to a particular gene, such as ERCC6 [3,2,8]. There is no database which takes gene name ERCC6 as input, and outputs the corresponding amino acids in the orthologous gene of non-human mammals. Instead, one needs to execute the following query plan. We first need to use gene name ERCC6 as input to query on an SNP database such as dbSNP to find all non-synonymous SNPs and their amino acid positions. Then, we, still taking ERCC6 as input, use a gene database, such as Entrez Gene, to obtain the encoded proteins in human species and other orthologous species. After that, using the proteins obtained from Entrez Gene, we search a sequence database to find the sequences. Finally, we use the sequences and the amino acid positions of the SNP obtained from dbSNP as input to do an alignment using an alignment database such as Entrez BLAST. Clearly, manually querying multiple data sources by filling multiple online query forms, keeping track of the obtained results and combining the results together is a tedious and error-prone process.

To address the above challenges, we have been developing a biological data integration system, SEEDEEP. This system enables exploring and querying of scientific deep web data sources [4]. The system infrastructure of SEEDEEP is shown in Figure 2. SEEDEEP comprises two components, which are the exploring and the querying components. Further, there are 6 individual modules, including *Schema Mining Module (SMi)*, *Schema Matching Module (SMA)*, *Query Planning Module (QP)*, *Query Reuse Module (QR)*, *Incremental Plan Generation Module (IPG)*, and *Plan Execution Module (PE)*. Given a set of deep web data sources, the exploring components (SMi and SMA) explore data sources to understand their usage and relationships. User queries are handled by the querying components (QP, QR, IPG and PE). The query planning module (QP),

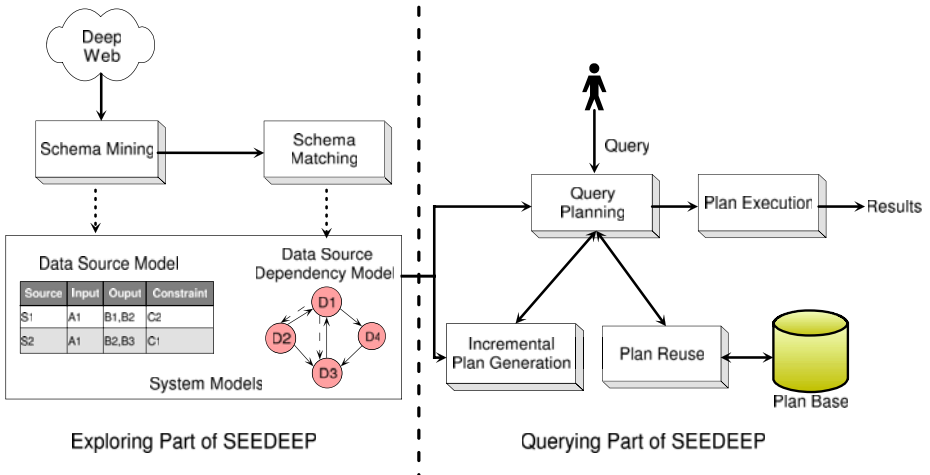


Fig. 2. System Infrastructure of SEEDEEP

working in conjunction with QR and IGP, finds the most appropriate query plan for a user query. Then, the query plan is executed by the plan execution module. The focus of this paper is the design and implementation of the plan execution module, with the goal of achieving efficiency through parallelism.

In our system, a query Q comprises n , $n > 1$, search terms, and it is formally denoted as $Q = \{t_1, t_2, \dots, t_n\}$. Among these n search terms, there is at least one search term, t_i , which is a concrete entity name. Such a term helps initiate answering of the query. For example, $Q1 = \{t_1 = ERCC6, t_2 = SNPID, t_3 = BLAST\}$ has three search terms, and, here, t_1 is the entity name.

Thus, a query plan can be modeled by a directed graph $G = (V, E)$, where the node set V represents the data sources involved in the plan, and the edge set E refers to the inter-dependence between these data sources. A directed edge $e(u, v) \in E$, where $u \in V, v \in V$, implies that an output attribute of the data source u can be used to query the data source v . The graph for the query plans in our system is a DAG (Directed acyclic graph), which implies that there are no cycles.

For an edge $e(u, v) \in E$, the data source u is the parent of the data source v , and inversely, the data source v is the child of the data source u . In query plans, it is possible that a data source may have multiple parents and/or multiple children. For a data source with multiple parents, each input query is composed of multiple attributes from its parents, and the input queries are constructed by combining all the instances from its parents. For example, if a data source $v \in V$ has n parents: (u_1, u_2, \dots, u_n) , and accordingly, the instance set from the parents are (I_1, I_2, \dots, I_n) , then the input query for v is (i_1, i_2, \dots, i_k) , where $i_j \in I_1 \cup I_2 \cup \dots \cup I_n$.

Figure 1 shows the query plan for the example query we had described earlier in this section. The query plan comprises four data sources: **dbSNP**, **Entrez Gene**, **Entrez Protein**, and **Entrez Blast**. The text on the edge refers to the attribute involved in the output-input relation between the data sources. We can see that **Entrez Gene** and **dbSNP** take the input *Gene* directly from the user. However, the data source **Entrez Protein** takes *protein ID*, which is an output from **Entrez Gene**. The input to **Entrez Blast** comprises two attributes: (*amino acid position*, *protein sequences*), which can be obtained from querying **dbSNP** and **Entrez Protein**, respectively.

The integration system executes the query plans automatically, submitting queries to multiple data sources. Our previous research shows that nearly 80% of the execution time is spent on data delivery between server and clients if the data sources involved in the query plan are visited in simple sequential order [28]. Thus, the efficiency of the keyword search is greatly influenced by how these queries are executed. The query plan could involve tens or hundreds of queries to be submitted to even a single data source. In order to save the time spent while waiting for results from data sources, we can query multiple data sources in parallel. For example, from the Figure 1, we can see that data sources **dbSNP** and **Entrez Gene** both take input from user's input and can be queried in parallel. The goal of the system described in this project is to effectively utilize the parallelism in a query plan, and querying multiple data sources in parallel.

3 System Design and Implementation

This section describes the design of the system for exploiting parallelism. Initially, we describe the different types of parallelism that we could target, and the challenges in exploiting them.

3.1 Parallelism in Query Plans

We can see that there are three types of parallelism that could be exploited during the execution of the query plans.

1. *Task Parallelism:* Data sources without inter-dependence can be queried in parallel. As we mentioned earlier, in the query plan shown in Figure 1, **dbSNP** and **Entrez Gene** have no edge in between, and they can be queried in parallel.

2. *Thread Parallelism:* Multiple queries can be submitted to the same data source, by issuing each query through a separate thread in parallel. This is because most deep web data sources support multiple queries simultaneously. For example, in Figure 1, using gene ERCC6 to query **Entrez Gene**, we can get multiple protein IDs, such as: NP_000115.1, NP_001100766.1, NP_001074690.1, etc. These multiple proteins IDs can be submitted to the data source **Entrez Protein** in parallel.

3. *Pipeline Parallelism:* The output of a data source can be processed by its child(ren) data source(s), while the data source can process new input queries. This is referred to as pipelined parallelism. For example, protein sequences output by **Entrez Protein** can be immediately used to query the data source **Entrez Blast**, while the data source **Entrez Protein** can process new protein IDs that are output by its parent node, the data source **Entrez Gene**.

While it appears that there is a significant opportunity to exploit parallelism, there are several challenges in doing so and still maintaining correctness. First, there is inter-dependence between the data sources, i.e., for an edge $e(u, v)$ in the query plan, the data source u should be queried before the data sources v . For example, in the query plan shown earlier in Figure 1, **Entrez Gene** provides protein ID that is used for querying the data source **Entrez Protein**, and thus, **Entrez Gene** should be queried before **Entrez Protein**. Second, data sources may have multiple parents and/or children, which makes tracking such dependencies more complex. If a data source has multiple parents, the data source should be queried after each of its parents. But, now suppose that these parents can be queried in parallel. In this case, since different data sources may have different response times, maintaining the inter-dependence with multiple parents while also exploiting task parallelism between the parents can be challenging. Furthermore, we may also want to exploit the pipelined parallelism between each of the parents and the child, and thread parallelism for the child.

3.2 System Details

We propose a novel system design that aims at exploiting the parallelism within a query plan, while still respecting the precedence constraints. In our design,

a *querier pool* and a *data pool* are built to implement the parallel query plan execution. The querier pool is composed of a set of queriers that all work in parallel. Queriers write results obtained from data sources into the data pool. Each querier corresponds to a data source. Its responsibilities include monitoring the activity at the parents of the data source, obtaining input instances, and using the obtained input instances to request data from the data source. It also interacts with the querier pool of the child(ren) data source(s), by sending a *start signal* and then the output results when the necessary inputs of the child(ren) are available. The data pool is used to store the results from data sources. It also facilitates the communication and synchronization between the queriers. Figure 3 shows the querier pool and data pool our system creates for the query plan we had shown earlier in Figure 1.

We now provide further details of the functioning of these two components.

Querier Pool: As we stated above, the querier pool comprises a querier corresponding to each data source. A querier is implemented as an interactive two-stage multi-threading system, which is shown in Figure 4. The first stage contains a *main thread*, which is used to implement the communication and synchronization with other queriers, including monitoring its parents to obtain the start signal, obtaining input instances, sending the start signal(s) to notify its child(ren) nodes, and forwarding output results to them. The second stage is composed of a set of querying threads which are initiated by the first stage. Each querying thread requests data from the data source and sends results back to the main thread.

The functioning for the main thread in the first stage is shown as Algorithm 3.1. In the first stage, a main thread is built to monitor the *start signals*, which are sent by the parents when they have new results to output.

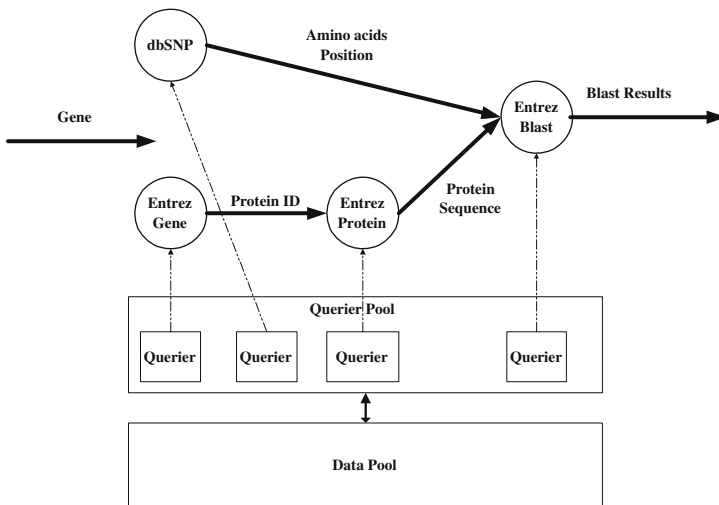


Fig. 3. An Example of System Architecture

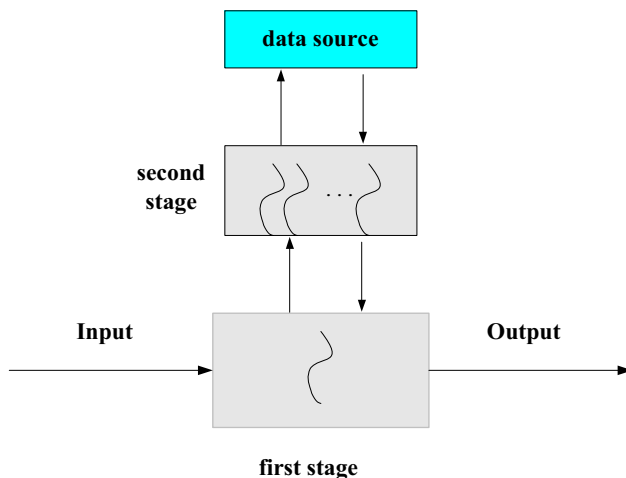


Fig. 4. Two Stage Multi-threading System

The main thread stays idle when there is no start signal from its parents. When it detects a start signal from its parents, it starts reading in new instances from its parents. The main thread constructs new input queries from these instances, launches a set of querying threads for querying the corresponding data sources and waits for the results. The set of querying threads launched by the main thread is in the second stage as shown in Figure 4. For data source with only a single parent, the new input queries are created directly from the new instances output by its parent. However, for data sources with more than one parent, the main thread sets local arrays to store all the instances from its parents. New input queries are constructed by combining the new instances of the parent with the old instances from other parents, which were stored in the local arrays. The new instances are also stored in the local arrays.

When the main thread receives result data from querying threads in the second stage, it will send start signals to its children and output the returned results. When the main thread receives data from all the querying threads in the second stage successively, it continues to check whether there are new start signals outputted by its parents, which will be stored in a buffer if the main thread is receiving data from the second stage. If the main thread discovers that there exists new instances, the querier will repeat the process of reading in the new instances and querying the data source again. Otherwise, if at least one querier for its parents is still working, the main thread waits to be awaked by start signals from its parents. The main thread will end if all the queriers of its parents end, and all querying threads it has created have returned results.

For a data source without parents, the task for the corresponding queriers becomes easier. For these data sources, the main threads take input directly from user's input, and then launches a set of querying threads for querying the corresponding data source.

Data Pool: The synchronization and communication between the main threads for data sources is supported by the data pool. All the main threads write results obtained from their corresponding data sources into the data pool. Once a main thread receives results from the corresponding data source and writes them into the data pool, it will notify all the main threads. Then the idle main threads would check whether there are new results coming from their parent nodes. If an awakened main thread discovers that the new results are from their parents, it reads new instances of the corresponding output results from the data pool, and uses them to query its data sources. Otherwise, the main thread will still stay idle. The data pool will have a lock with it to keep the synchronization between the read and write operation on the data pool. This lock ensures that only one main thread can access the data pool at the same time, which maintains the consistency of the data pool.

Algorithm 3.1: Querier(u, E)

```

/* $u$  is a data source,  $E$  is the edge set in the query plan*/
Launch a main thread  $T_m$  for  $u$ 
Parent set  $PS = \{v_1, v_2, \dots, v_n\}$ , where  $v_i$  has edge  $e(v_i, u) \in E$ 
Children set  $CS = \{w_1, w_2, \dots, w_m\}$ , where  $w$  has edge  $e(u, w_j) \in E$ 
if  $|PS| = 0$  /* $u$  has no parent*/
  take input  $DI$  from user's input
  launch querying threads to request data from  $u$ 
  while results are returned from querying threads of  $u$ 
     $u$  sends start signal and output results to  $w_j \in CS$ 
else
  if  $|PS| = 1$  /* $u$  has only one parent*/
    while querier for  $v_1 \in PS$  is still working
      while  $u$  receives a start signal
         $u$  reads new instances  $NI$  from  $v_1$ 's output
        launch querying threads to request data from  $u$ 
        while there are results returned from querying threads of  $u$ 
           $u$  sends start signal and output results to  $w_j \in CS$ 
      if  $v_1$  is still working
         $u$  waits for start signals
    else /* $u$  has multiple parents*/
      Set local arrays  $LA$  to store instances from output of  $v_i \in PS$ 
      while there is querier for  $v_i \in PS$  is still working
        while  $u$  receives a start signal from data source  $v_i$ 
           $u$  reads new instances  $NI$  from  $v_i$ 's output
          construct input query sets  $IS$  by combining  $NI$  and  $LA$ 
          launch querying threads to request data from  $u$ 
           $u$  stores  $NI$  in local arrays  $LA$ 
          while there are results returned from querying threads of  $u$ 
             $u$  sends start signal and output results to  $w_j \in CS$ 
      if there is querier for  $v_i \in PS$  is still working
         $u$  waits for start signals

```

3.3 Exploiting Parallelism

The system design we have developed is able to exploit the three kinds of parallelism we had mentioned earlier.

Task Parallelism: In our system, data sources are queried by queriers in the querier pool. Since queriers only detect the start signal and output results from their parents, queriers' execution is only impacted by their parents, i.e., queriers for data sources without inter-dependence can work in parallel. Further, by detecting the start signals sent by their parents, queriers will wait until they get new instances for constructing new queries, which implies that the parents are queried before the children. In this way, the precedence constraints are respected in our algorithm. For a data source with multiple parents, the querier will detect the start signals from all the parents. In this way, the dependence between multiple parents and data source is also respected. For a data source with multiple children, it would notify all the children when the corresponding querier obtains new results. Accordingly, queriers for the children would get data from the data pool and work in parallel.

Thread Parallelism: Exploiting thread parallelization is straightforward in our system. Multiple input queries constructed by the main thread are submitted to the data source in parallel by a set of querying threads. These querying threads work in parallel.

Pipeline Parallelism: In the system, the start signal is sent by queriers immediately after obtaining new results. The start signal informs the child(ren) data source(s) that they can read new results from their parents and process new queries. At the same time, the queriers sending the start signal can continue to get new results from their own parents, and process new data. For data sources with multiple parents, the queriers set local arrays for storing all the instances coming from their parents, and thus maintain the integrity of input queries for querying the data source.

4 Implementation and Evaluation Study

The performance of our system is evaluated using keyword queries that execute on 28 biological deep web data sources, including *SNP 500*, *Entrez Gene*, *Entrez Protein*, *Entrez Blast*, *Alfred*, and others. More details of this integration system are available from our earlier publications [3,2,8]. In the experiments we conducted, 30 query plans are constructed by submitting queries to the query planning module of the system [3]. The queries, whose form are described in Section 2, are specified by a domain expert we have collaborated with.

Due to changing network condition and work load of the servers, the response time for the data sources we have integrated varies to some extent over time. Thus, we perform all experiments during the same time of the day. In addition, the experiments are performed three times and the final results we report are the average of the three execution times.

Table 1 shows the time spent on constructing query plans and the times spent on executing these query plans *sequentially* and in parallel for the 30 queries. In

Table 1. Sequential and Parallel Execution Times

Query ID	# of Attr.	# of Data Sources	Plan Time(s)	Seq Time(s)	Parallel Time(s)
1	1	1	0.193	0.738	0.697
2	1	2	0.110	2.690	1.106
3	1	1	0.104	6.880	7.434
4	1	1	0.105	6.775	5.881
5	1	2	0.120	9.756	10.552
6	2	1	0.115	3.770	4.482
7	1	2	0.105	0.961	0.899
8	1	4	0.113	163.429	13.813
9	5	2	0.115	6.292	6.432
10	1	1	0.110	11.445	13.182
11	7	3	0.120	15.986	9.637
12	8	5	0.110	19.686	12.171
13	9	5	0.109	19.300	7.324
14	13	6	0.115	44.035	12.651
15	11	7	0.125	32.281	7.049
16	9	4	0.110	29.029	14.510
17	12	5	0.119	30.803	11.413
18	10	5	0.125	28.874	11.354
19	16	5	0.110	21.552	7.181
20	20	7	0.148	120.306	17.288
21	18	8	0.109	123.079	17.096
22	20	4	0.109	30.974	12.008
23	21	6	0.110	16.997	11.467
24	23	6	0.104	25.453	14.179
25	36	12	0.110	128.313	18.803
26	24	7	0.109	45.030	5.706
27	34	10	0.119	48.601	13.046
28	26	9	0.120	37.128	13.061
29	40	11	0.110	131.418	20.113
30	38	11	0.125	47.259	11.477

the sequential implementation, only one data source is queried at one time, thus neither of task, thread, or pipelined parallelism are used. In the table, the first column corresponds to the query ID of each query. We also show the number of attributes contained in each query in the second column and the number of data sources involved in the query plans in the third column. We can notice that some queries in our experiments have a very large number of attributes, such as more than 20. The reason is that the user may not use the exact attribute terms (*schema-level keywords*) used in data source schemas, instead, they use some high-level abstract names (*semantic-level keywords*). One semantic level keyword usually can be mapped to multiple schema-level keywords, such as a semantic level keyword "SNP_Frequency" can be mapped to 4 schema level keywords (Population, Sample, Allele.Frequency and Genotype.Frequency). The attributes counted in Table 1 are the schema level keywords which are obtained

from user’s semantic level keywords (usually smaller than 5) using a domain ontology. The table shows that the time spent on executing query plans is much larger than time spent on constructing query plans, which demonstrates the importance of utilizing parallel query plan execution. In addition, the table clearly shows that for most queries, parallel execution results in better response times, though the speedup factor varies significantly. The maximum speedup is by a factor of 12, whereas 6 of the queries have a speedup better than 6. We will further analyze this later in the section.

One important observation here is that for queries 3, 5, 6, 9 and 10, the simple sequential approach execution has better performance. This is because there is very little parallelism in these queries, which is also correlated with the small number of attributes or data sources contained in the query plans. Further, because of the communication and synchronization overheads, the implementation reported in this paper results in a small slowdown for these queries.

4.1 Detailed Analysis

We now analyze how the speedup achieved on a query is impacted by the number of data sources and attributes. We also study the relative impact of task parallelism as well as thread and pipelined parallelism on the overall speedups.

Our discussion will use the metric *Speedup Ratio*, which is computed as $\frac{t_{seq}}{t_{para}}$. Here, t_{seq} is the time spent executing the query plan sequentially and t_{para} is the time spent executing the query plan using our system.

Figure 5 shows the speedups of different queries as a function of the number of attributes involved in the query. First, we can see that most query plans have a speedup ratio greater than 1, implying that we can exploit parallelism effectively. We can also see a general trend that the speedup increases with the number of attributes. This is because a query with a larger number of attributes is likely to have a more complex query plan, with more potential for parallel execution. However, the number of attributes does not absolutely determine the amount of

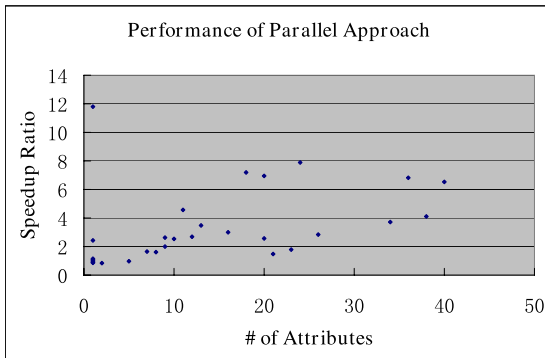


Fig. 5. Speedups Vs. No. of Attributes in the Query

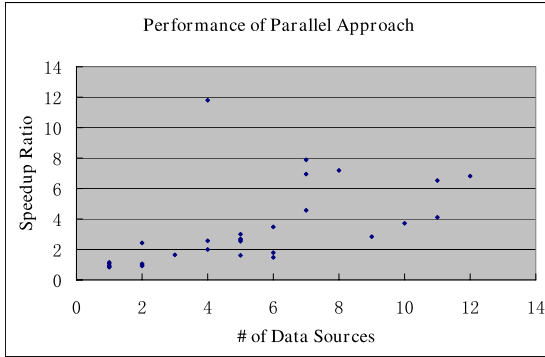


Fig. 6. Speedup Vs. No. of Data Sources

parallelism. We can also see that the speedup ratio can be quite large even with small number of attributes.

Figure 6 shows the speedups as a function of the number of data sources involved in query plans. There is again a general trend in the figure that increasing the number of data sources increases the speedup. This is to be expected, as a large number of data sources could imply more task parallelism. However, similar to what we saw with the number of attributes, the correlation is not always very strong.

Our next step was to further understand how the different types of parallelism contribute to the overall speedups. For this purpose, we built a *Task-Parallel* system that only exploits task parallelism in query plans. In this system, data sources without inter-dependence can be queried in parallel. However, the input queries obtained from parents are submitted to data sources sequentially. Figure 7 shows the speedups - both the overall speedups with all three types of parallelism, as well as the speedup from the Task-Parallel system. As

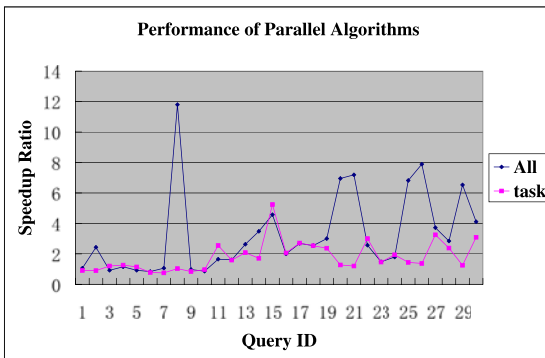


Fig. 7. Speedups: Different Types of Parallelism

the implementations of thread and pipelined parallelism are closely tied to each other, we just focused on isolating the gains from task parallelism.

From Figure 7, we can see that both task parallelism as well as the combination of thread and pipelined parallelism is important for our system. While the Task-Parallel system has better speedups than sequential system, the system with all three types of parallelism still has better speedups.

From the Figure 7, we can learn that, for queries 8, 20, 21, 25, 26, and 29, the system with all three forms of parallelism has much better speedups than the Task-Parallel system, implying that thread and pipelined parallelism is more important for these. By analyzing these query plans, we discover that, compared with other plans, these query plans involve at least some data sources which have a large number of input instances. These queries are mainly about discovering amino acids occurring at the corresponding position in the orthologous gene of non-human mammals with respect to a particular gene, or discovering SNP and SNP related information, such as ssID, about a particular Gene. Taking discovering SNP related information as an example, a particular Gene may have multiple SNPs, and a SNP may have multiple ssIDs. Thus, in these query plans, data sources, like dbSNP, have a large number of input queries(instances). Thus, the potential for thread and pipelined parallelism is very high. On the contrary, for other query plans where thread and pipelined parallelism has little improvement over the task-parallel system, the number of input queries(instances) for the data sources is limited. Another observation from Figure 7 is that, for some query plans, the Task-Parallel system has better performance compared with the system with all parallelization. This is because there is no potential for thread and pipelined parallelism, whereas communication and synchronization between additional threads results in a slow-down.

Overall, we can see that for most queries, there is a significant benefit from exploiting parallelism. Certain queries have a higher potential for task parallelism. These queries typically involve a large number of data sources and/or attributes, and thus, are more likely to have data sources without inter-dependence. Some other queries can benefit more from thread and pipelined parallelism. This is because of the larger number of input queries to each data source.

5 Related Work

Parallelization has previously been recognized as an effective approach to improving the efficiency of deep web data source query processing. Warnick *et al.* have developed a query engine to access heterogenous data resources on the deep web in parallel [9]. However, this system does not consider the possibility of inter-dependence between data sources, and therefore, the implementation is much simpler. Braga *et al.* [10] present a parallel framework for multi-domain queries on the Web that constructs multiple query plans for the same query, and then selects the optimal one according to cost metrics. While the system exploits task parallelism in executing query plans, it does not consider thread or pipeline parallelism. Bruno *et al.* [11] have studied thread parallelization in processing

top-k queries over web-accessible databases. Deshpande *et al.* [12] propose flow algorithms using pipelined parallelism to solve a multi-way join query problem. Multiple query plans are used simultaneously to minimize the processing time of a query. They focus on utilizing the parallelization between multiple query plans.

Our work is also related to Web Service Management System (WSMS) [13], which is proposed for solving the problem of Select-Project-Join queries spanning multiple web services. They *join* the results from web services in the query plan at the end to obtain their final answer. Thus, the cost of the query plans in their work mainly depends on the order of the web services. Our work is clearly different, as the queries we are handling are keyword queries. We do not *join* results, but *merge* results from different data sources in the query plan. Thus, exploiting the parallelism among data sources is the core of the optimization in our case.

Our work is also closely related to Triana Workflows System [14]. In the Triana system, distributed workflows, which are provided by users, dynamically map to the virtual grid overlay of triana services. In our work, the query plan is constructed by a query planning system rather than user input.

Parallelization is also utilized in large, distributed web search systems, which are based on distributed indexes over multiple servers. Parallel query processing is required in the search system due to the distributed index and processors. Rasolofo *et al.* [15] propose a collection selection and result merging strategy for distributed Web search system. Orlando *et al.* [16] have developed a distributed Web search engine, MOSE, which utilizes two types of parallelism, task parallelism and data parallelism, in processing queries. Thus, our work aims at processing queries by requesting data from multiple data sources, whereas these efforts aim at processing queries by using multiple processors.

Parallelization is widely used for faster query processing in traditional relational databases [17,18,19]. Besides task parallelism and pipeline parallelism, they also exploit data parallelism, which is based on partitioning of the data. Thread parallelization is not exploited in query processing in traditional relational databases, since these systems do not request data from distributed web sources. In addition, because of the nature of the queries and data sources, our system design is very different.

6 Conclusions

This paper has described and evaluated a system that exploits parallelization for accelerating search over multiple deep web data sources. An interactive, two-stage multi-threading system design enables us to achieve task parallelization, thread parallelization, and pipelined parallelization. This system design has been evaluated in the context of a biological integration system that targets SNP related data. Our experimental study used 30 queries over 28 data sources. Our results show that the benefit from parallelization varies significantly. The maximum speedup we obtain is 12, and 6 of the queries achieve a speedup of 6 or

better. Certain queries have a higher potential for task parallelism. These queries typically involve a large number of data sources and/or attributes, and thus, are more likely to have data sources without inter-dependence. Some other queries can benefit more from thread and pipelined parallelism. This is because of the larger number of input queries to each data source.

Acknowledgements

This work was supported by NSF grants 0541058, 0619041, and 0833101. The equipment used for the experiments reported here was purchased under the grant 0403342.

References

1. Babu, P., Boddepalli, R., Lakshmi, V., Rao, G.: Dod: Database of databases—updated molecular biology databases. *Silico. Biol.* 5 (2005)
2. Wang, F., Agrawal, G., Jin, R., Piontkivska, H.: Snpminer: A domain-specific deep web mining tool. In: *Proceedings of the 7th IEEE International Conference on Bioinformatics and Bioengineering*, pp. 192–199 (2007)
3. Wang, F., Agrawal, G., Jin, R.: Query planning for searching inter-dependent deep-web databases. In: Ludäscher, B., Mamoulis, N. (eds.) *SSDBM 2008*. LNCS, vol. 5069, pp. 24–41. Springer, Heidelberg (2008)
4. Wang, F., Agrawal, G.: Seeddeep: A system for exploring and enquiring scientific deep web data sources. In: *Proceedings of SSDBM 2009* (2009) (to appear)
5. He, B., Zhang, Z., Chang, K.C.C.: Knocking the door to the deep web: Integrating web query interfaces. In: *Proceedings of the 2004 ACM SIGMOD international conference on Management of Data*, pp. 913–914 (2004)
6. Chang, K., He, B., Zhang, Z.: Toward large scale integration: Building a metaquerier over databases on the web (2005)
7. He, H., Meng, W., Yu, C., Wu, Z.: Automatic integration of web search interfaces with wise_integrator. *The international Journal on Very Large Data Bases* 12, 256–273 (2004)
8. Wang, F., Agrawal, G., Jin, R.: A system for relational keyword searches over deep web data sources. Technical Report OSU-CISRC-03/08-TR10, The Ohio State University (March 2008)
9. Warnick, W.L., Lederman, A., Scott, R.L., Spence, K.J., Johnson, L.A., Allen, V.S.: Searching the deep web: Directed query engine applications at the department of energy. Technical report (2001)
10. Braga, D., Ceri, S., Daniel, F., Martinenghi, D.: Optimization of multi-domain queries on the web. In: *Proceedings of VLDB 2008*, pp. 562–573 (2008)
11. Bruno, N., Gravano, L., Marian, A.: Evaluating top-k queries over web-accessible databases. In: *ICDE*, p. 2004 (2002)
12. Deshpande, A., Hellerstein, L.: Flow algorithms for parallel query optimization. In: *IEEE 24th International Conference on Data Engineering*, 2008. *ICDE 2008*, pp. 754–763 (2008)
13. Srivastava, U., Munagala, K., Widom, J., Motwani, R.: Query optimization over web services. In: *VLDB 2006: Proceedings of the 32nd international conference on Very large data bases, VLDB Endowment*, pp. 355–366 (2006)

14. Churches, D., Gombas, G., Harrison, A., Maassen, J., Robinson, C., Shields, M., Taylor, I., Wang, I.: Programming Scientific and Distributed Workflow with Triana Services. *Concurrency and Computation: Practice and Experience (Special Issue: Workflow in Grid Systems)* 18(10), 1021–1037 (2006)
15. Rasolofo, Y.: Approaches to collection selection and results merging for distributed information retrieval. In: *CIKM*, pp. 191–198 (2001)
16. Orlando, S., Perego, R., Silvestri, F.: Design of a parallel and distributed web search engine. In: *Proceedings of Parallel Computing (ParCo) 2001 conference*, pp. 197–204. College Press, Imperial (2001)
17. Chaudhuri, S.: An overview of query optimization in relational systems. In: *PODS, AC*, pp. 34–43 (1998)
18. Hong, W., Stonebraker, M.: Optimization of parallel query execution plans in xprs. Technical Report UCB/ERL M91/50, EECS Department. University of California, Berkeley (1991)
19. Hasan, W.: Optimization of sql queries for parallel machines. PhD thesis, Stanford University (1995)

A Visual Interface for on-the-fly Biological Database Integration and Workflow Design Using VizBuilder*

Shahriyar Hossain and Hasan Jamil

Integration Informatics Laboratory, Department of Computer Science
Wayne State University, Michigan, USA
shah_h@wayne.edu, jamil@cs.wayne.edu

Abstract. Data integration plays a major role in modern Life Sciences research primarily because required resources are geographically distributed across continents and experts depend upon leveraging these digitally archived resources. The ever changing and exponentially growing digital archives pose a significant challenge for traditional data integration efforts and efficient development of data processing pipelines for scientific investigations. Thus a forceful argument can be made that in Life Sciences fast and ad hoc application design requiring data integration and workflow processing is urgently warranted. In this paper, we present a new visual application design toolkit, called *VizBuilder*, that can be used to design ad hoc applications at throw away costs by naive users in the Life Sciences to integrate remote databases and to design application pipelines. We leverage the recent development of *BioFlow* query language and *LifeDB* database management system that allows for such application design. We show that *VizBuilder* can be used as a front end query interface for *LifeDB*, with which robust and a wide range of applications can be developed at a very abstract and conceptual level without having to learn *BioFlow*. Users are able to express their application by drawing it using *VizBuilder* icons and connecting them in a meaningful way. Once completed, *VizBuilder* can compile and transform the visual application into an error-free *BioFlow* program to be executed in *LifeDB*. We present the functionality of *VizBuilder* using a real life application that our Life Sciences researchers have implemented and executed in *LifeDB* as part of their investigation.

1 Introduction

Despite remarkable progress in computer technology, the task of developing programs has remained a writing exercise. Although modern compilers help programmers with automatic code generation, interface building etc., they operate on the premise that the developer has a good command over the textual programming language. In most cases, text based approach makes more sense than

* This research was partially supported by National Science Foundation grants CNS 0521454 and IIS 0612203.

the graphical alternative, as it allows developers to have fine control over complex control flows and data structures. Textual programs are easy to maintain and share. The situation is changing with the introduction of high level languages in emerging fields such as data integration, spatial data analysis, business workflow management, and so on. The user demographics of these languages include people from disparate scientific fields, user bases in e-commerce societies, first responders in disaster mitigation systems, etc. To facilitate access and querying, most applications and web based services develop form and GUI centered interfaces. Such static form based approaches are bound to fall short of capturing the full capabilities of these powerful languages. On the other hand, even a skilled developer finds it hard to learn the textual syntax of every other language. Visual programming system seems to be the most logical solution to these problems. At the Integration Informatics Laboratory, we are working on a language for integration of hidden web resources. To support ad hoc and on-the-fly data integration in Life Sciences, we are developing a new query language called BioFlow for our biological data management system, LifeDB. Our experience with our Life Sciences users and colleagues shows ample evidence that even though BioFlow offers high level abstractions for programming support, developing applications involving diverse internet resources is still a challenge for them. On the other hand, depicting a program with diagrams, flow charts and sequence of conceptual constructs was more user friendly and appealing. So, we set ourselves towards building a graphical editor called VizBuilder as a visual toolkit that will ultimately generate BioFlow scripts from application description diagrams for execution by LifeDB engine.

Our principal goals in the design of VizBuilder were to adopt simplicity of design, support conceptual application development at a higher abstraction level than currently possible, and follow best practices in workflow systems [14,21]. The design was inspired by several academic [20,22,28] and industrial [2] workflow systems. We preferred independence from the underlying computational structures such as grid [19], or web services [3] for wider acceptance and applicability. Unlike systems such as [31,29,2], we have also preferred a decoupled environment as opposed to a tight coupling of the interface and execution engine so that modular design and leveraging existing systems such as LifeDB are possible. In summary, we use VizBuilder to visually express an application at a conceptual level. Then a translation mechanism is used to convert the visual specification to a BioFlow script for onward execution by the LifeDB engine. Although BioFlow is largely declarative, it contains programming constructs such as loops and conditions, which is a departure from [9]. VizBuilder is also *accessible* in a way similar to BioGuideSRS [8] and Ali Baba [24] requiring no installation. Finally, from a design standpoint, VizBuilder extends the ideas of graph grammars in visual languages such as [31,13,30,29] by enriching them with features from *domain specific modeling* to support abstraction and decoupling from lower level engines.

The remainder of the paper is organized as follows. In section 2, we introduce salient features of BioFlow on intuitive grounds using a real life data integration

miRNA	chromosome	microRNA	geneName	pValue
hsa-mir-10a	ch 17	hsa-mir-10a	FLJ36874	0.004
hsa-mir-205	ch 1	hsa-miR-196b	MYO16	0.009

(a) genes

geneID	miRNA	targetSites	pValue
FLJ36874	hsa-mir-10a	10	0.004
FLJ36874	hsa-mir-10b	3	null
RUNDC2C	hsa-mir-205	8	null
MYO16	hsa-miR-196b	null	0.009

(b) sangerRegulation

geneID	miRNA	targetSites	pValue
FLJ36874	hsa-mir-10a	10	0.004
FLJ36874	hsa-mir-10b	3	null
RUNDC2C	hsa-mir-205	8	null
MYO16	hsa-miR-196b	null	0.009

(e) regulation

geneID	miRNA	targetSites	pValue	p63Binding
FLJ36874	hsa-mir-10a	10	0.004	Y
FLJ36874	hsa-mir-10b	3	null	Y
RUNDC2C	hsa-mir-205	8	null	Y
MYO16	hsa-miR-196b	null	0.009	N

(c) micrornaRegulation

Gene	p63Binding
FLJ36874	Y
RUNDC2C	Y
MYO16	N

(d) proteinCodingGene

geneID	miRNA	targetSites	pValue	p63Binding
FLJ36874	hsa-mir-10a	10	0.004	Y
FLJ36874	hsa-mir-10b	3	null	Y
RUNDC2C	hsa-mir-205	8	null	Y
MYO16	hsa-miR-196b	null	0.009	N

(f) proteinCodingGeneRegulation

Fig. 1. User tables and data collected from microRNA.org and microRNA.sanger.ac.uk

application in micro RNA data analysis. This introduction will be leveraged in section 3 where we discuss the system architecture of VizBuilder, and subsequently discuss application building using VizBuilder in section 4. We will discuss the graph grammar we used in VizBuilder for BioFlow language and show the reconstruction of the BioFlow script from section 2 with VizBuilder visual operators to convince the reader of the flexibility of our system. In section 5, we discuss where VizBuilder stands in relation to other similar systems and in data integration, generally. Finally, we conclude in section 6 with our after thoughts and plans for the future research.

2 A Motivating Application: BioFlow by Example

To illustrate the capabilities of BioFlow, we adapt a real life Life Sciences application discussed in [15] which has been used as a use case for many other systems and as such can be considered a benchmark application for data integration. A substantial amount of glue codes were written to implement the application in [15] by manually reconciling the source schema to filter and extract information of interest. Our goal in this section is to show how simple and efficient it is to develop this application in LifeDB.

In this example, the user wants to validate the hypothesis that “*the human p63 transcription factor indirectly regulates certain target mRNAs via direct regulation of miRNAs*” by submitting several queries in multiple different internet sites and combining information from them in a specific way. If positive, the user also wants to know the list of miRNAs that indirectly *regulate* other target mRNAs with high enough confidence score (i.e., $pValue \leq 0.0006$ and $targetSites \geq 2$), and so he proceeds as follows. He collects 52 genes along with their chromosomal locations (shown partially in figure 1(a) as the table *genes*) from a wet lab experiment using the host miRNA genes and map at or near genomic p63 binding sites in the human cervical carcinoma cell line ME180. He also has a set of several thousand direct and indirect proteincoding genes (shown partially in figure 1(d) as the table *proteinCodingGenes*) which are the targets of p63 in ME180 as candidates. The rest of the exploration thus proceeds as follows.

He first collects a set of genes (*geneIDs*) for each of the miRNAs in the table *genes*, from the web site www.microRNA.org by submitting one gene at a time in

the form that returns for each such gene, a set of gene names that are known to be targets for that miRNA. The site returns the response in the form of a table from which the user collects the *targetSites* alongwith the gene name partially shown as the table *micrornaRegulation* in figure 1(c).

To be certain, he also collects the set of gene names for each miRNA in table *genes* from *microrna.sanger.ac.uk* in a similar fashion partially shown in table *sangerRegulation* in figure 1(b). Notice that this time the column *targetSites* is not available, and so he collects the *pValue* values. Also note that the scheme for each of the tables are syntactically heterogeneous, but semantically they are similar (i.e., *miRNA*≡*microRNA*, *geneName*≡*geneID*, and so on). He does so because the data in the two databases are not identical, and there is a chance that querying only one site may not return all possible responses. Once these two tables are collected, he then takes a union of these two sets of gene names (in *micrornaRegulation* and *sangerRegulation*), and finally selects the genes from the intersection of the tables *proteinCodingGenes* (that bind to p63, i.e., *p63Binding*=‘N’) and *micrornaRegulation*∪*sangerRegulation* as his response.

To compute his answers in BioFlow using LifeDB, all he will need to do is execute the following script that completely implements the application. It is interesting to note that in this application, the total number of data manipulation statements used are only seven (statements numbered (2) through (8)). The rest of the statements are data definition statements needed in any solution using any other system. We will describe shortly what these data manipulation sentences mean in this context. For now, a short and intuitive explanation is in order while we refer interested readers to [17] for a more complete exposition.

In program 1, the statements numbered (1) through (7) are most interesting and unique to BioFlow. The **define function** statements essentially declare an interface to the web sites at URLs in the respective from clauses, i.e., *microrna.org* and *microrna.sanger.ac.uk*. The **extract** clause specifies what columns are of interest when the results of computation from the sites are available, whereas the **submit** clauses say what inputs need to be submitted. In these statements, it is not necessary that the users supply the exact variable names at the web site, or in the database. The wrapper (**FastWrap** [6]) and the matcher (**OntoMatch** [7]) named in the **using** clause and available in the named ontology *mirnaOntology*, actually establish the needed schema correspondence and the extraction rules needed to identify the results in the response page. Essentially, the **define function** statement acts as an interface between LifeDB and the web sites used in the applications.

To invoke and compute at these sites, we use **call** statements at (4) and (5). The first statement calls **getMiRNA** for every tuple in table *genes*, while the second call only sends one tuple to **getMiRNASanger** to collect the results in tables *micrornaRegulation* and *sangerRegulation*. The statements (6) and (7) are also new in BioFlow. They capture respectively the concepts of *vertical* and *horizontal* integration in the literature. The **combine** statement collects objects from multiple tables possibly having conflicting schemes into one table. To do so, it also uses a key identifier (such as **gordian** [27]) to recognize objects

Program 1. A BioFlow script for integrating heterogeneous data sources

```
process compute_mirna { (1)
  open database bioflow_mirna;
  drop table if exists genes;
  create datatable genes {
    chromosome varchar(20), start int, end int, miRNA varchar(20) };
  drop table if exists proteinCodingGene;
  create datatable proteinCodingGene {
    Gene varchar(200), p63binding varchar(20) };
  drop table if exists micrornaRegulation;
  create datatable micrornaRegulation {
    mirna varchar(200), targetsites varchar(200), geneID varchar(300) };
  define function getMiRNA
    extract mirna varchar(100), targetsites varchar(200), geneID varchar(300)
    using wrapper mirnaWrapper in ontology mirnaOntology
    from "http://www.microrna.org/microrna/getTargets.do"
    submit(matureName varchar(100), organism varchar(300));
  drop table if exists sangerRegulation;
  create datatable sangerRegulation {
    microRNA varchar(200), geneName varchar(200), pvalue varchar(200) };
  define function getMiRNASanger
    extract microRNA varchar(200), geneName varchar(200), pvalue varchar(30)
    using wrapper mirnaWrapper in ontology mirnaOntology
    from "http://microrna.sanger.ac.uk/cgi-bin/targets/v5/hit_list.pl/"
    submit(mirna_id varchar(300), genome_id varchar(100));
  load data local infile '/genes.txt' (2)
    into table genes fields terminated by '\t'
    lines terminated by '\r\n';
  load data local infile '/proteinCodingGene.txt' (3)
    into table proteinCodingGenes fields terminated by '\t'
    lines terminated by '\r\n';
  insert into micrornaRegulation
    call getMiRNA with select miRNA, '9606' from genes ; (4)
  insert into sangerRegulation
    call getMiRNASanger with select miRNA, '2964' from genes ; (5)
  insert into regulation (combine micrornaRegulation, sangerRegulation
    using matcher OntoMatch identifier gordian); (6)
  insert into proteinCodingGeneRegulation
    (link regulation, proteinCodingGene
    using matcher OntoMatch identifier gordian); (7)
  select *
    from proteinCodingGeneRegulation where p63binding='N'; (8)
  close database bioflow_mirna; }
```

across tables. Such concepts have been investigated in the literature under the titles record linkage or object identification. For the purpose of this example, we adapted GORDIAN [27] as one of the key identifiers in BioFlow. The purpose of using a key identifier is to recognize the fields in the constituent relations that essentially make up the object key¹, so that we can avoid collecting non-unique objects in the result. The `link` statement, on the other hand extends an object in a way similar to join operation in relational algebra. Here too, the schema matcher and the key identifier play an important role. Finally, the whole script can be stored as a named *process* and reused using BioFlow's `perform` statement. In this example, line (1) shows that this process is named `compute_mirna` and can be stored as such for later use. We are now ready to introduce VizBuilder which can be used to develop this entire application using its visual constructs and operators.

3 VizBuilder System Overview

As shown in figure 2, the VizBuilder system is divided into two major components – the editor and the kernel. The editor has several predefined features such as, the capability to *open*, *close* or *save* an editing session and *undo/redo* operations. The language specific operators are not part of the original system. Consequently, each language specific feature needs to be embodied in the icons as a combination of forms, and a series of defined steps, so that, once followed, VizBuilder can translate these interactions and collected descriptions into statements in the target language. So, during the *deployment phase* of VizBuilder, a language designer needs to define the visual language as an instance of the eXtensible VizBuilder Markup Language (XVML). The kernel then converts the XVML definition to a set of VizBuilder specific Java classes and objects. The tool bar of the editor is reconstituted from the visual alphabet of the target XVML instance. A set of *syntax directed editing* rules are also implanted into the editor. At this stage, the kernel derives a translation scheme from the visual model of VizBuilder to the textual language. The system is then recompiled with these components and deployed in the web for programming purposes.

In the *programming phase*, the end user opens up the editor in the browser and draws the visual program by dragging the operators and the edges from the toolbar². The *syntax directed editing* rules, decided in the deployment phase, guides the workflow development with on the fly syntax checking, for example, denying a connection between operator A and operator B, if any such connection is prohibited in the rule base. When the user chooses to compile his or her workflow, it is checked for syntactic and semantic errors. If the program is error free then it is translated into the code of the target textual language using the *model2code* translation scheme set in the deployment phase.

¹ Note that object key in this case is not necessarily the primary keys of the participating relations.

² We will take a closer look at a programming session in section 3.2.

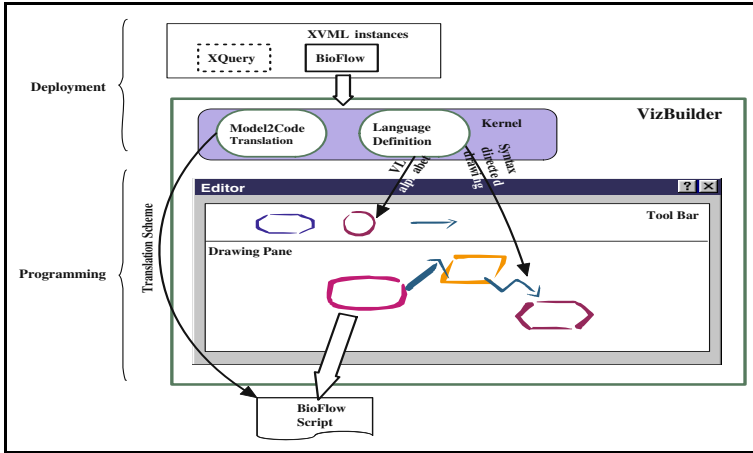


Fig. 2. System Overview of VizBuilder

From the implementation point of view, VizBuilder is developed in Java which makes it operating system independent. It is an applet which can be deployed with standard web servers. So, the end user just needs a Java enabled browser like Internet Explorer, Mozilla-Firefox, Safari or Opera to draw a visual program with VizBuilder. We used Java Universal Network/Graph Framework (JUNG) [1] as the library for drawing and maintaining graphical structures.

3.1 Operator Definition Using eXtensible Vizbuilder Markup Language (XVML)

VizBuilder captures semantics of its visual constructs expressed in XVML – a markup language designed after XAML [4]. In this section we will present a flavor of XVML because a complete exposition is outside the scope of this paper. Interested readers may refer to [16] for a more thorough treatment of the underlying foundation of VizBuilder. Similar to the *Code-behind* mechanism in XAML, we allow the `Class` attribute to link to Java classes. Thus, we are able to create the user interface elements declaratively, while hiding the imperative logic for syntax checking and translation inside the class definitions. These classes are extended from the base class `Operator` which is embedded in VizBuilder. The overridden abstract functions in these classes ensure orderly translation from the visual model to the textual program. The `id` attributes are unique throughout the definition to avoid ambiguity of reference.

As shown in definition 3.1, every instance of XVML starts with an `Operator` node. Each operator must have a child node from one of two possible types, namely, the `NestedPanel` or the `InputPanel`. A `NestedPanel` holds a number of other operators in its toolbar. `NestedPanels` come with a default drawing area. For example, the `Process` operator in definition 3.1 is the root node of any visual program. The root operator does not appear physically in the VizBuilder. It is the `NestedPanel`, embedded inside the `Process` operator, that the end user will see

Definition 1. Partial definition of BioFlow with XVML

```

<Operator id="Process" Class="Process.java" icon="process.jpg">
  <NestedPanel>
    <Operator id="Write" Class="Write.java" icon="write.jpg">
      <NestedPanel>
        <Operator id="Table" Class="Table.java" icon="table.jpg">
          <InputPanel>
            </TextBox id="TableName" label="Table Name">
              ...
            </InputPanel>
          </Operator>
        <Operator id="Function" Class="Function.java" icon="function.jpg">
          ...
        </Operator>
      </NestedPanel>
    </Operator>
  ...
  <Operator id="Assign" Class="Assign.java" icon="assign.jpg">
    <InputPanel>
      </TextBox id="assignLeft" label="left">
      </TextBox id="assignRight" label="right">
    </InputPanel>
  </Operator>
</NestedPanel>
</Operator>

```

when she opens up her editor. This `NestedPanel` holds a number of operators in its toolbar, two of them (`Write` and `Assign`) are shown in definition 3.1. The recursive chain of *operator-panel* ends with an operator with an `InputPanel`, as the `InputPanels` are not allowed to contain any `Operator` or drawing pane. For example, if `VizBuilder` is assembled following definition 3.1 then the top two layers, (`Process` and `Write`) will allow workflow design, while the `Table` layer will only support textual input. We support varieties of GUI element like text boxes, combo boxes, lists etc. During the translation process, the overridden *translate* functions in the linked classes are called in a bottom up fashion. The function converts the workflow drawn on the `NestedPanel` or the user inputs from the `InputPanel` into the script. The *translate* functions tackle the syntactic errors by throwing an exception. The exception is caught at the topmost layer (`Process`) and a descriptive message with a link to the erroneous operation is shown to the user. The semantic errors are handled in the server side. Based on the server response, an alert message similar to the one for syntactic error is generated.

3.2 A Sample VizBuilder Application Design Session

In figure 3, we have three layers of panels marked by three numbered circles. Layer (1) is the `NestedPanel` for the `Process` operator from definition 3.1. The

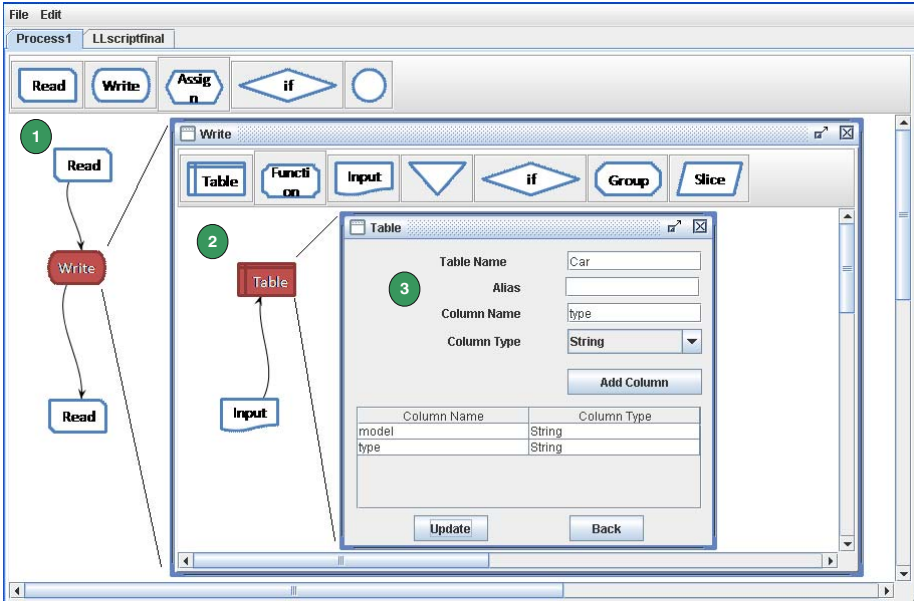


Fig. 3. A sample programming session

user has dragged down three operators from the toolbar and drawn a visual statement. One of the operators, namely, the **Write** is expanded in layer (2). This layer is also of the **NestedPanel** type; allowing the user to draw workflows. The user has drawn a workflow where data from a disk file is uploaded to a database table. Layer (3) is different from the previous two layers. It is an **InputPanel** describing the schema of the table. The hierarchy of the panels makes programming relatively straightforward. For example, if the user switches to one of the **Read** operators in layer (1), all the nested windows for the **Write** operator will disappear. The user can work on multiple processes simultaneously by switching between the tabs containing them. Another important point is that we never run out of real estate or impose any ordering scheme. The user is free to draw the graph with as many operators and place them in a manner that is aesthetically pleasing to her.

4 BioFlow with VizBuilder

Although it is possible to use VizBuilder for different programming languages by changing the XVML instances, our primary target was BioFlow. The statements in BioFlow can be divided into three categories, namely - control, data definition and data manipulation statements. **Processes**, like `comput_mirna` in program 1, are independent programs written in BioFlow. The tables and functions defined inside a **Process** have global scope. The tables can be manipulated by the two

basic data manipulation statements - **Select** and **Insert** which add on to the functionalities of their SQL counterparts. Apart from the tables, BioFlow allows the use of local variables as temporary storages. It supports conditional branching and loop to control the flow of execution.

4.1 Process Panel

The root node of the XVML definition of BioFlow is the **Process** operator (definition 3.1). The flow of execution within a process is drawn on the **NestedPanel** of this operator. The operators in this layer are the **Read**, **Write**, **Assign**, **Condition** and the end of block (**EOB**). The first two operators have one to one relationship with the **Select** and **Insert** statements in BioFlow. The **Assign** operator is responsible for accessing the local variables, while the remaining two operators define the control blocks. We don't have any visual operator for the data definition statements in the process layer. They are handled in subsequent nested layers. Three types of edges are allowed in this layer. Figure 4(a) describes the production rules for the grammar for the **Process** panel with the terminal symbols are shaded. The *Stmt* node is linked to the *Read* or *Write* statement in BioFlow. The edges represent the flow of execution. The edges out of the **If** node are labeled as *t* and *f*. They portray conditional branching.

In the XVML definition of the **Process** layer, every child operator except the **EOB** has a child panel describing that operator. For the **Assign** and the **Condition** operators, these panels are of type **InputPanel**. The **Read** and **Write** operators, on the other hand, have their own **NestedPanels** with different graph grammars. We will describe them in the following section.

4.2 Read and Write Panels

These panels describe the flow of data in different directions, unlike the control flows in the **Process** panel. In a **Write** statement, data can travel from the

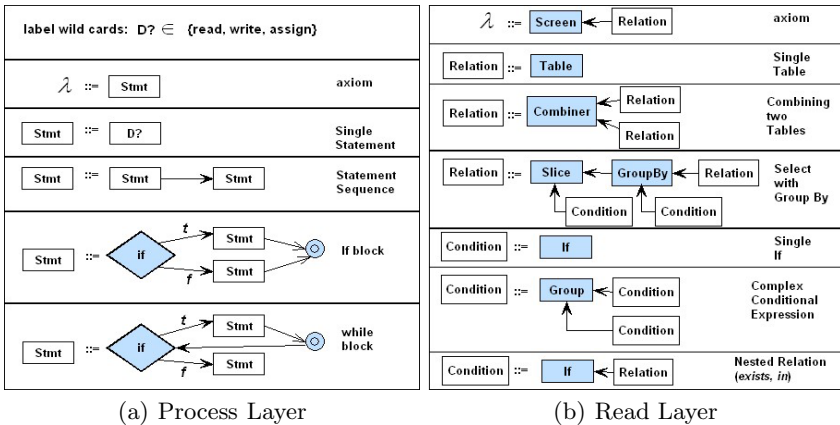
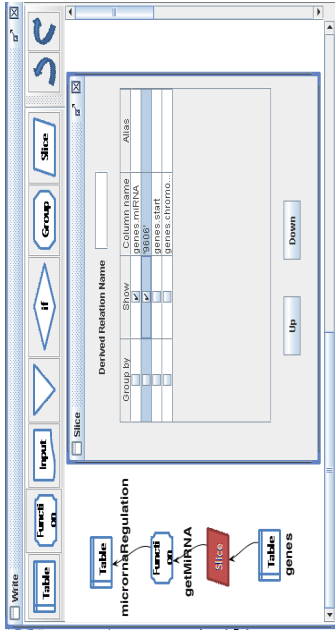


Fig. 4. Graph Grammars for the two nested layers in visual BioFlow

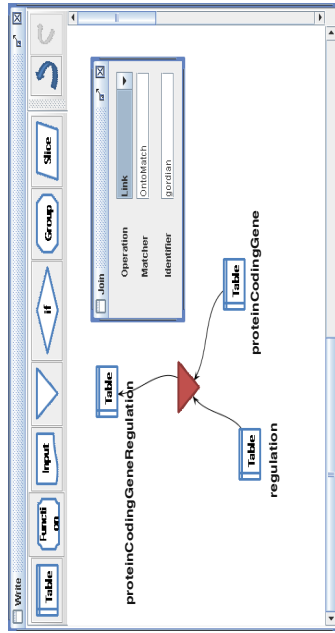
input devices, disk storages or source tables, through a number of intermediate tables and modifying functions, to a destination table. On the other hand, the **Read** statements extract data from one or more tables; modify them to desired formats by applying different operators and display them on output devices. The following set of operators is employed to express the data flow in the **Read** and **Write** panels.

- **Table** operator denotes the schema of the data table in consideration. An outgoing edge from the **Table** node holds the relation that is stored in the table. An incoming edge to a **Table** node describes one or more rows that will be inserted into the table. One does not need to define the schema every time a table is accessed. The user can choose from a pool of already defined schemas, instead. This ensures the global scope of BioFlow.
- **Function** operator represents the callable functions in BioFlow. Its use is restricted to the **Write** operations only. An outgoing edge from a **Function** node holds the resultant relation from a function call. The incoming edges provide the input parameters to a function.
- **Input** operator can be connected to a **Table** or a **Function** node with an outgoing edge. It collects data from the user through the input devices. The **Input** operator can be used to extract raw data from a file.
- **Combiner** operator takes in multiple relations and transforms them into one using one of the available algorithms. Currently, we support natural join of SQL along with BioFlow specific functions - **link** and **combine** as possible merging algorithms.
- **Slice** operator changes the input relation by renaming or discarding one or more columns of the input schema. This operator can also change the values of every tuple of a column by applying some arithmetic function. In other words, **Slice** is analogous to the **projection** clause in relational algebra.
- **Condition** operator can be used to build the **where** clause in an **insert** or a **select** statement. This operator can only be connected to a **Slice** or a **Group** node by an outgoing edge for proper functioning. It trims out the rows that fail the logical condition set by the operator.
- **Group** operator can be connected to a **Slice** operator with an outgoing edge. **Group** operator helps realize complex conditions of the form *((condition 1 and condition 2) or condition3)*.
- **Screen** operator can be used within a **Read** statement only. It cannot have any outgoing edge. It is allowed to have a single incoming edge from a **Table**, **Function**, **Slice** or a **Combiner** operator. It projects the incoming relation onto the screen.

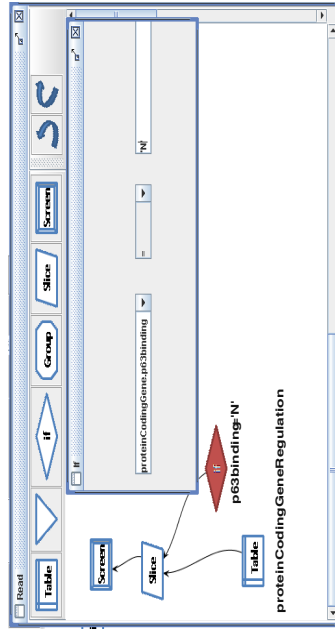
The graph grammar for the **Read** panel is shown in figure 4(b). The grammar for the **Write** panel is similar to this one. Again, the terminal symbols are shaded.



(b) *micromiRNARegulation* is populated with *getMiRNA* (4)



(c) Linking table *regulation* and *proteinCodingGene* (7)



(d) Final *Read* with a condition (8)

Fig. 5. Partial miRNA application script development using VizBuilder (The indices following the subtitles are the line numbers of program I.)

4.3 Revisiting miRNA Regulation: The Running Example

Armed with our understanding of VizBuilder and the visual language for BioFlow, let us revisit the `compute_mirna` process from section 2. In figure 5, four representative statements from the seven data manipulation statements are shown. Figure 5(a) holds the first `load` statements, where data from `'/genes.txt'` are uploaded into `genes` table. The process layer with all seven data manipulation statements is also shown in this figure. The schema of the `genes` table is shown as a case of `InputPanel` at work. Figure 5(b) corresponds to the statement (4) of program 1. Here, we have data flowing from the table `genes` to a web page encapsulated by the web function `getMiRNA`. The result of the function call is stored in the `micrornaRegulation` table for later use. The `InputPanel` of the `Slice` operator displays the SQL projection of the two columns, namely `miRNA` and `'9606'`. Figure 5(c) demonstrates the use of the `Combiner` operator in statement (7) of program 1. It links the tables `regulation` and `proteinCodingGene`. Finally, in figure 5(d) we have statement (8) of program 1. The content of the table `proteinCodingGeneRegulation` is displayed to the user after performing a SQL selection over the rows based on the condition `p63Binding='N'`.

5 Related Research

The LifeDB system features several state of the art components that represent a wide area of research in data integration and scientific workflow management. In this paper, we are addressing the issue of visual application design using a graphical tool, such as VizBuilder, and mostly used the workflow and data integration aspects of its query language BioFlow. Our goal was to introduce an effective alternative to textual programming as an aid for application development in Life Sciences where data integration and workflow design is commonplace. We believe the foregoing discussion has made it amply clear that BioFlow allows us to support visual data integration and workflow design in VizBuilder. Nonetheless, it should also be apparent that the way VizBuilder is designed, it can be used to support application design in any other declarative language, and as such VizBuilder stands on its own as a stand alone generic interface for visual application builder. In this section, we will, however, compare VizBuilder with other similar tools that support data integration in Life Sciences to understand where it stands compared to those systems.

There are several leading contemporary systems in the literature such as Taverna [22], BioGuide [8], HyperFlow [12], Kepler [5], and Kaleidoquery [23]. Taverna is a web service based data integration and workflow management system. Unlike Taverna, BioFlow can be used to extract data from plain html pages, and application design does not require source site cooperation. Taverna models the integration problem through functional programming. As a result, it lacks the power of declarative manipulation of data which is achieved in the `read` and `write` panels of BioFlow, for example. Taverna's editor also has a fixed real state allocated for drawing. So, large workflows with more than ten processors are

difficult to comprehend. BioGuide on the other hand is a biological information integration system with excellent usefulness. However, the precomputed path based approach of querying database resources is not flexible enough compared to SQL based BioFlow where ad hoc integration with arbitrary resources is supported. In contrast, the data sources to be supported by BioGuide are integrated and embedded in the system making it difficult to allow querying new resources needed for a given user application. In simple terms, it only allows join, selection and projection type of queries on the supported databases. In contrast, VizBuilder supports arbitrary queries using arbitrary databases and local applications, local or remote, in a way similar to Taverna, but with significant differences. Interested readers may refer to [18] for a comprehensive comparative exposition of Taverna and BioFlow. Another recent development is HyperFlow which combines visual workflow language with visual query language. Despite its interesting set of features, it is not possible to design complex control flows with HyperFlow. The Kepler workflow system supports scientists from different domains including biology, ecology and astrophysics in which users are allowed to construct workflows using local applications as well as external sources. However, one needs to manually wrap and add a resource to Kepler’s library of actors before it can be used in a workflow.

As BioFlow is a superset of SQL, its visual representation can be treated as a visual query language (VQL) system. A comprehensive survey of the VQL systems can be found in [10]. It is very difficult for a user without any database knowledge to use the Entity Relationship based VQLs like QBD* [25] and VKQL [26]. DFQL [11] is a dataflow based query language where a large number of operations such as complex Boolean predicates, aggregate functions, and grouping, are done textually. It is interesting to note that the visual language of BioFlow shares many constructs with Kaleidoquery. However, the 3D representation of Kaleidoquery requires a significant amount of CPU power without obvious gains in utility. Also, its join, self-join and nested query construct are less intuitive than visual BioFlow.

6 Summary and Future Research

Visual programming is the next logical step in the realm of computer programming. In this paper, we proposed a generic editor that can be tuned to fit a large array of programming languages. Our main objective was to make programming interesting to non-programmers through eye-catching icons, intuitive drawing rules, easy web access etc. We are planning to take it one step beyond. Currently, a designer needs to write the XVML instances and extend the class hierarchy manually during the deployment phase. We want to automate these steps and build a “*super-editor*” where the customization can be carried out graphically. We are also developing a rigorous representation of the nested graph grammar and testing it with representative programming languages from different paradigms. The current VivBuilder version does not show the results after each step in execution; only the final outcome is displayed. The simplest

solution to this problem would be to show all the intermediate data tables created during the workflow run. But, this gives rise to some difficulties with loops and conditional statements in the workflow. Currently, we allow textual results from the workflows to be displayed in a tabular format. We will gradually allow graphical information visualization techniques such as images, Dendogram, heat maps, networks etc. As a somewhat separate but parallel project, we are working on the optimization of the workflows drawn by the naive user.

References

1. JUNG - Java Universal Network/Graph Framework, <http://jung.sourceforge.net>
2. Microsoft Robotics, <http://msdn.microsoft.com/en-us/library/bb483088.aspx>
3. Web Services Description Language (WSDL) Version 2.0., <http://www.w3.org/TR/wsd120/>
4. XAML - Extensible Application Markup Language, <http://msdn.microsoft.com/en-us/library/ms752059.aspx>
5. Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludäscher, B., Mock, S.: Kepler: An extensible system for design and execution of scientific workflows. In: SSDBM, pp. 21–23 (2004)
6. Amin, M.S., Jamil, H.: FastWrap: An Efficient wrapper for Tabular Data Extraction from the Web. In: Tenth IEEE International Conference on Information Reuse and Integration, Las Vegas, Nevada (August 2009)
7. Bhattacharjee, A., Jamil, H.: OntoMatch: A Monotonically Improving Schema Matching System for Autonomous Data Integration. In: Tenth IEEE International Conference on Information Reuse and Integration, Las Vegas, Nevada (August 2009)
8. Boulakia, S.C., Biton, O., Davidson, S.B., Froidevaux, C.: Bioguidesrs: querying multiple sources with a user-centric perspective. *Bioinformatics* 23(10), 1301–1303 (2007)
9. Bowers, S., Ludäscher, B., Ngu, A.H.H., Critchlow, T.: Enabling scientific workflow reuse through structured composition of dataflow and control-flow. In: Proceedings of the 22nd International Conference on Data Engineering Workshops, Washington, DC, USA, p. 70. IEEE Computer Society Press, Los Alamitos (2006)
10. Catarci, T., Costabile, M.F., Levialdi, S., Batini, C.: Visual query systems for databases: A survey. *Journal of Visual Languages & Computing* 8(2), 215–260 (1997)
11. Dogru, S., Rajan, V., Rieck, K., Slagle, J.R., Tjan, B.S., Wang, Y.: A graphical data flow language for retrieval, analysis, and visualisation of a scientific database. *Journal of Visual Languages and Computing*, 247–265 (1996)
12. Dotan, D., Pinter, R.Y.: HyperFlow: An integrated visual query and dataflow language for end-user information analysis. In: Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing, Washington, DC, USA, pp. 27–34 (2005)
13. Ehrig, K., Ermel, C., Hänsgen, S., Taentzer, G.: Generation of visual editors as eclipse plug-ins. In: Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering, pp. 134–143 (2005)

14. Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., Myers, J.: Examining the challenges of scientific workflows. *Computer* 40(12), 24–32 (2007)
15. Gusfield, D., Stoye, J.: Relationships between p63 binding, dna sequence, transcription activity, and biological function in human cells. *Mol. Cell* 24(4), 593–602 (2006)
16. Hossain, S., Jamil, H.: VizBuilder – a generic editor for visual languages. Technical report, Computer Science, Wayne State University (December 2008)
17. Jamil, H., El-Hajj-Diab, B.: Bioflow: A web-based declarative workflow language for life sciences. In: *IEEE International Workshop on Scientific Workflow*, July 2008, pp. 453–460 (2008)
18. Jamil, H., Islam, A.: The power of declarative languages: a comparative exposition of scientific workflow design using BioFlow and Taverna. In: *IEEE International Workshop on Scientific Workflow*, Los Angeles, CA (2009)
19. Kesselman, C., Foster, I.: *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, San Francisco (1998)
20. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y.: Scientific workflow management and the kepler system. *Concurrency and Computation: Practice & Experience* 18(10), 1039–1065 (2006)
21. McPhillips, T., Bowers, S., Zinn, D., Ludäscher, B.: Scientific workflow design for mere mortals. In: *Future Generation Computer Systems* (2008)
22. Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M., Wipat, A., Li, P.: Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics* 20(17), 3045–3054 (2004)
23. Paton, N.M.N.: Kaleidoquery: A visual query language for object databases. In: *Advanced Visual Interfaces*, pp. 247–257. ACM Press, New York (1998)
24. Plake, C., Schiemann, T., Pankalla, M., Hakenberg, J., Leser, U.: Alibaba: Pubmed as a graph. *Bioinformatics* 22(19), 2444–2445 (2006)
25. Santucci, G., Sottile, P.A.: Query by diagram: a visual environment for querying databases. *Software - Practice and Experience* 23(3), 317–340 (1993)
26. Siau, K., Chan, H., Tan, K.: Visual knowledge query language as a front-end to relational systems. In: *Proceedings of the Fifteenth Annual International on Computer Software and Applications*, September 1991, pp. 373–378 (1991)
27. Sismanis, Y., Brown, P., Haas, P.J., Reinwald, B.: GORDIAN: efficient and scalable discovery of composite keys. In: *VLDB*, pp. 691–702 (2006)
28. Taylor, I., Shields, M., Wang, I., Harrison, A.: *The Triana Workflow Environment: Architecture and Applications*. In: Taylor, I., Deelman, E., Gannon, D., Shields, M. (eds.) *Workflows for e-Science*, pp. 320–339. Springer, New York (2007)
29. Tolvanen, J.-P., Rossi, M.: Metaedit+: defining and using domain-specific modeling languages and code generators. In: *Companion of the 18th annual ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications*, pp. 92–93 (2003)
30. Vangheluwe, H., de Lara, J.: Computer automated multi-paradigm modelling for analysis and design of traffic networks. In: *Proceedings of the 36th conference on Winter simulation*, pp. 249–258 (2004)
31. Zhang, D.-Q., Zhang, K.: VisPro: A visual language generation toolset. In: *Proceedings of the IEEE Symposium on Visual Languages*, Washington, DC, p. 195 (1998)

EpiC: A Resource for Integrating Information and Analyses to Enable Selection of Epitopes for Antibody Based Experiments

Niall Haslam and Toby Gibson

European Molecular Biology Laboratory
Structural and Computational Biology Unit
Heidelberg, Germany
haslam@embl.de

Abstract. EpiC is a web resource for choosing epitopes for use in antibody based experiments. It integrates a suite of tools to identify the subsequence of a target protein that a binder interacts with. Where the binder is an antibody this subsequence is termed an epitope, i.e. the region of a protein that an antibody specifically recognises. EpiC is part of the Europe-wide project ProteomeBinders, that aims to develop a resource of binding molecules for the entire Human Proteome. EpiC has a simple interactive web interface that provides experimentalists with an integrated suite of bioinformatic tools and annotation information. Crucially this neither requires the user to download any specialist software nor develop scripting tools.

Given the diversity of applications for affinity reagents, it is important that the resource provides relevant information dependent on the type of experiment being carried out. It is important to note that researchers often require multiple different binders for one protein and different binders for use in different experiments. EpiC achieves access to these wide range of tools using state of the art methods of data integration. Webservices and REST interfaces such as the Distributed Annotation System are used to deliver the information to the user. These ensure that the latest versions of the analysis software and annotation are displayed to the user, relieving the user of the need to keep up to date with the latest databases and software.

1 Introduction

Affinity reagents play a crucial role in a wide range of biological applications. The ability to specifically identify proteins in a mixture is of vital importance to researchers [1]. The latest data from UniProt suggests that there are around 24,000 human proteins [2]. This number provides a lower limit on the number of affinity reagents that will be required to identify the complete human proteome. Currently, the most widely used affinity reagents are antibodies; they are the first choice of experimentalists when designing affinity based experiments. They can specifically bind to a target protein by recognising a subsequence of the protein

known as the epitope. A set of binding reagents that could be used to comprehensively explore the human proteome would be of great use to the biological community. In addition to individual proteins reagents will be required for all the splice variants, polymorphisms and peptides of these proteins. Post-translational modifications, such as glycosylation, as well as modifying the affinity of an antibody for the target could alter the selection of epitope for a given experiment. This will raise the number of reagents required to well over 10 times the number of proteins. Furthermore, the choice of experiment will impact on the selection of the epitope. The optimal epitope for a western blot could be different to that for an ELISA experiment. If the epitope contains a potential glycosylation site then antibodies can be used to distinguish between the glycosylated and non-glycosylated states; i.e. one would be targeted and specific to the glycosylated state and one to the non-glycosylated state. The availability of reagents is a limiting factor to the current exploration of the human proteome. The development of tools to aid in the identification of epitopes and the generation of antibodies will greatly increase the rate at which such reagents can be synthesised.

1.1 Rationale for Project

The use of bioinformatic tools in the generation of a set of epitopes for the generation of antibodies will be key to achieving the goal of a set of affinity reagents for the human proteome [1]. We present in this paper our work to unify the tools and annotations that are necessary for experimentalists to select appropriate epitopes to drive the generation of antibodies. Currently, the tools for helping experimentalists select epitopes are spread around a number of sites or interfaces [3]. By integrating these tools for biologists, in an easy to use and modular fashion we are providing a modern bioinformatics platform for affinity proteomics and other fields that use antibodies. Resources such as the Human Protein Atlas [4], have collated a resource of well characterised affinity reagents for a wide range of experimental contexts. Given the scale of the problem, bioinformatics resources that speed up the process of selecting epitopes are required. Novel approaches, such as the antibodypedia, which attempts to build community sourced repository of data on antibodies have been launched recently [5].

Proteins are composed of functional modules that range in size from small linear motifs of a few amino acids to globular domains that compose the entire protein. Experimental investigation of the protein could potentially require binders to each module. Therefore there has to be an awareness of the domain composition of the protein. We use a range of tools to determine this, SMART, dispred and Pfam [6], [7] [8]. Further information about the secondary structural components that these domains are composed of are returned via sources such as DSSP and UniProt annotations [9] [2]. The investigation of functional aspects of a protein can have the effect of limiting the choice of epitope that an experimentalist can use. For example, if an experimentalist is seeking to disrupt the phosphorylation of a particular protein at a specific motif, then there is only one epitope that can be selected [10]. If localisation is the focus of study then the experimentalist will need to know where the localisation signals on the protein

are. Finally, the epitope needs to be unique in order to ensure the specificity of the antibody. Therefore once the features of the protein have been displayed it is crucial to be able to determine their similarity on the primary sequence level to features of other proteins. For this, we use the BLAST server at the EBI [11].

1.2 Why Go for Distributed Data Collection

The rise of distributed or federated systems of retrieving biological information has gained widespread adoption in the bioinformatics community because it removes the requirement to maintain large and expensive databases by multiple groups. Such systems facilitate the creation of added-value to the original data, using it as a springboard to develop further functionality to the resources without the costs of maintenance. The standardisation of data formats has enabled the creation of a new generation of tools and software that build upon the primary data repositories such as UniProt and Ensembl. Webservice technologies have also gained a lot of attention in bioinformatics in the last few years [11]. They offer a simple way for programmers to interact with a heterogeneous range of technologies from Grid Storage Services to high performance computers upon which to store and initiate jobs [12]. These do not require the programmer to have an in depth knowledge of the architecture of the grid or the computers which they interact with [13]. This abstraction frees the developer to concentrate on the application logic of their software. Perhaps more importantly it means that relatively low powered machines can be used as gateway servers that connect and manage sophisticated bioinformatic analysis on a large number of distributed servers and services [14]. It would not be feasible for a single developer or small team to maintain a suite of servers to run a wide range of software tools, but it is feasible to connect via webservices to these tools and run them remotely. For this reason, the webservices on offer at institutes like the EBI have seen massive adoption in the previous years [11].

2 Methods

EpiC is composed of two main parts. First, the user interface which is deployed as a javascript application in the user's browser. This allows the user to interact with the application logic which is run in a tomcat server. The user interface acts as a client to the server through remote procedural calls (RPC). This is shown schematically in Figure 1. EpiC requires the user to answer some simple questions about the type of experiment being carried out. It contains a set of analysis modules, which can be considered as distinct sets of bioinformatic analysis tools. For example, if the experimenter is designing an experiment that requires a phospho-epitope then the phosphorylation prediction would be activated. In this way, the resource is capable of delivering context-dependent analysis to the user. This is in contrast to the currently existing bioinformatic infrastructures which are fragmented and require the user to run disparate bioinformatic analysis tools.

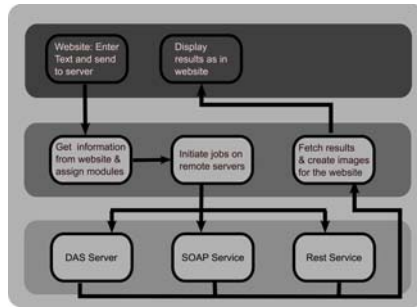


Fig. 1. Representation of how the user (top) interacts and receives information via the javascript interface, from the application server of the resource (middle). Crucially all of the interactions with remote computing infrastructure (bottom) are hidden from the user.

2.1 Distributed Annotation System and Webservices

The federated biological data retrieval system DAS is essentially a set of servers that provide information about, in our case, proteins. The servers provide this data in a protocol that is the same between servers [15]. EpiC acts as a client collating relevant information about the protein of interest and displaying it to users. In addition to experimentally determined annotations about a protein, it is possible to collate pre-calculated bioinformatic analysis about the protein [15]. This increases the speed of response since there is no need to recalculate results for each query. DAS provides methods for ensuring that the version of the protein in the reference server and the annotation server is the same. This allows the distribution of information between multiple servers and the subsequent integration by clients at a later stage. The precise amount of data and the different sources on display is left to the client [15].

We have written a number of clients to popular bioinformatics webservices, such as EMBOSS, BLAST and Protein Identifier Cross-Reference (PICR) services [16], [11], [17] and display this information in conjunction with the information retrieved via DAS. The application server acts as a client to these services and controls the running of jobs for the user. As soon as the results from these services are returned they are parsed into a format for display to the user on the website.

The use of an ajax based interface improves the usability for the user. Interfaces such as the Dasty client have demonstrated the utility of this approach [18]. This approach enables an interactive exploration of the protein, that would be difficult to achieve in other development models. The always up-to-date nature of webservices frees the user to concentrate on the biological problem rather than the problem of getting the most current information on their protein. This allows us to, for example, dynamically highlight sections of the graphs and update information about the protein as it becomes available.

2.2 Protein Affinity Reagent Ontology

EpiC makes use of the Protein Affinity Reagent (PAR) ontology to formally describe the reagents and the experimental contexts. The PAR ontology is an extension of the Protein Standards Initiative's Molecular Interaction ontology [19]. This allows EpiC to determine the intentions of the user in a formalised way. The calls to the ontology, made through the Ontology Lookup Service (OLS) at the EBI [20], are important in helping the user to understand the technical questions about the state of the protein, features and experimental context. Terms used in the webpage are queriable by the user and an informative popup can display the definition of the terms. When appropriate EpiC links to other ontologies, for example the Sequence Ontology), to describe more generic biological terminology. Such calls are made from the EpiC resource to the OLS at the EBI through webservices. This aids the user in identifying important parts of the protein for use in the selection of appropriate epitopes.

2.3 From Webpage to Application

The user is asked first of all to enter in a sub-sequence of a protein or the UniProt identifier of the protein that they are interested in. This could be a fragment that they are planning to use to raise an antibody. The PICR service from the EBI is used to identify the protein sequence and fetch its cross references in the other databases, such as UniProt and Ensembl [17]. They are then invited to describe in more detail the specifics about the intention of the experiment. These questions aim to determine which software modules should be run by the application server, which receives this data via an RPC call from the client. Once this is received the resource initiates appropriate jobs on the remote servers and collects the relevant annotation information. Figure 2 shows this schematically.

Information can be retrieved and integrated in the output from a wide range of biological sources. Information about the protein, for example, domains and

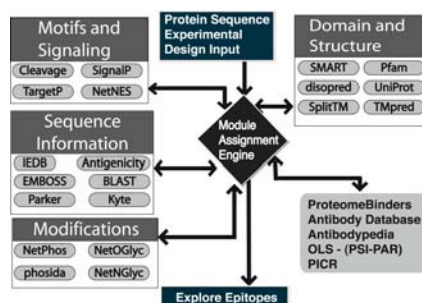


Fig. 2. A brief overview of some of the modules that the EpiC resource is capable of interacting with. A complete list is available on the website (<http://epic.embl.de/servers.html>).

structure [6], post-translational modifications [21], [22], [23] and membrane topology, [24], [25] can be selected for inclusion in the results on the basis of the experimental context. The module assignment engine currently only accepts jobs through the RPC calls, though we plan to open this up to other forms of webservice technology such as WSDL. This will aid in the reuse of the software in other pipelines and in software such as Taverna [14].

3 Results: Exploring the Epitopes

The user is able to access the tools and information from a single webpage that is dynamically updated. Bringing these tools together in one place and standardising the display of the results, from different sources in different data formats, represents a significant contribution to antibody development. Once the user clicks the run button, the answers to the questions are collated and used to determine the appropriate software analyses to be run. EpiC runs these jobs (See Figure 1), and returns the results as quickly as possible to increase the feeling of interactivity and speed. The client script in the user's browser then displays the results in one page (See Figure 3). The results are organised into logical groups to facilitate the exploration of the protein to determine the optimal epitopes. It is worth underlining that since there is not one definitive answer to the question it is impossible to serve the user with a list of predicted epitopes. To do so, may mislead the user and hinder the selection of appropriate epitopes for their experiments.

The results page summarises the relevant information about the protein. It begins with information about the physicochemical properties of the protein such as hydrophobicity and hydrophilicity [16]. These are commonly used by experimentalists to determine the relative antigenicity of the different regions of the protein. In addition traditional antigenicity scales, supplied by the Immune Epitope Database, are also shown for the same reason [3]. When appropriate, surface exposure and flexibility scales are also shown to highlight regions of the protein that are likely

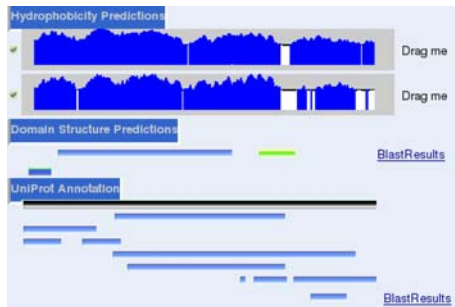


Fig. 3. A subset of the results showing the some of the possible analysis that can be run. The user is free to collapse and reorder results to make it even easier to align the analysis.

to be exposed and therefore available to a binder. General information about the protein, such as the gene ontology terms associated with its entry in UniProt are also shown to remind the user of the localisation of the protein or its position in the cell cycle. Below this is, if available, a link out to the Protein Atlas. The Human Protein Atlas is a compendium of characterised antibody reagents. Highlighting the availability of known antibodies to the user is important to ensure that they don't waste time recreating these reagents.

4 Discussion

We have presented a tool for the integration of bioinformatic tools and data for the use of experimentalists designing antibody based experiments. This resource incorporates a wide range of tools and information, within an easy to use interface. The web-based technologies used in this project enhance the usability of the resource, making the resource more user-friendly and increasing the utility of the individual tools by aggregating information.

The software is also made available as a facebook application, (<http://apps.facebook.com/epicsocial>), to demonstrate the possibilities of delivering software through javascript in a social networking environment. The development of EpiC has demonstrated the utility of distributed computing in conjunction with interactive webscripting technologies to deliver complex bioinformatic tools with no barriers to use for users. Leveraging distributed architectures with modern web design technologies has enabled us to add value to the services and tools that we build upon. Without frameworks such as DAS and webservice the completion of this project would have taken much longer and required many more developers. This resource demonstrates the utility of building middle-ware services such as DAS and webservice. We have created a useful resource that integrates a wide range of information in a coherent and intuitive manner that will be used by biologists to design the next generation of proteomics experiments.

Acknowledgements

This work was funded by the ProteomeBinders Consortium through the EU Framework 6 programme. We would also like to acknowledge the work of the biosapiens project, the EMBRACE network and the Distributed Annotation System.

References

1. Taussig, M.J., Stoevesandt, O., Borrebaeck, C.A.K., Bradbury, A.R., Cahill, D., Cambillau, C., de Daruvar, A., Dübel, S., Eichler, J., Frank, R., Gibson, T.J., Gloriam, D., Gold, L., Herberg, F.W., Hermjakob, H., Hoheisel, J.D., Joos, T.O., Kallioniemi, O., Koegl, M., Konthur, Z., Korn, B., Kremmer, E., Krobitch, S., Landegren, U., van der Maarel, S., McCafferty, J., Muyldermans, S., Nygren, P.-r., Palcy, S., Plückthun, A., Polc, B., Przybylski, M., Saviranta, P., Sawyer, A., Sherman, D.J., Skerra, A., Templin, M., Ueffing, M., Uhlén, M.: Proteomebinders: planning a european resource of affinity reagents for analysis of the human proteome. *Nature Methods* 4(1), 13–17 (2007)

2. Boutet, E., Lieberherr, D., Tognolli, M., Schneider, M., Bairoch, A.: Uniprotkb/swiss-prot. *Methods Mol. Biol.* 406, 89–112 (2007)
3. Zhang, Q., Wang, P., Kim, Y., Haste-Andersen, P., Beaver, J., Bourne, P.E., Bui, H.H., Buus, S., Frankild, S., Greenbaum, J., Lund, O., Lundegaard, C., Nielsen, M., Ponomarenko, J., Sette, A., Zhu, Z., Peters, B.: Immune epitope database analysis resource (IEDB-AR). *Nucl. Acids. Res.* 36 (July 2008)
4. Persson, A., Hober, S., Uhlén, M.: A human protein atlas based on antibody proteomics. *Curr. Opin. Mol. Ther.* 8, 185–190 (2006)
5. Björling, E., Uhlén, M.: Antibodypedia, a portal for sharing antibody and antigen validation data. *Mol. Cell. Proteomics* 7, 2028–2037 (2008)
6. Letunic, I., Doerks, T., Bork, P.: Smart 6: recent updates and new developments. *Nucl. Acids Res.* 37 (January 2009)
7. Ward, J.J., McGuffin, L.J., Bryson, K., Buxton, B.F., Jones, D.T.: The dispred server for the prediction of protein disorder. *Bioinformatics* 20, 2138–2139 (2004)
8. Finn, R.D., Tate, J., Mistry, J., Coggill, P.C., Sammut, S.J., Hotz, H.R., Ceric, G., Forslund, K., Eddy, S.R., Sonnhammer, E.L., Bateman, A.: The pfam protein families database. *Nucl. Acids. Res.* 36 (January 2008)
9. Kabsch, W., Sander, C.: Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* 22, 2577–2637 (1983)
10. Diella, F., Haslam, N., Chica, C., Budd, A., Michael, S., Brown, N.P., Trave, G., Gibson, T.J.: Understanding eukaryotic linear motifs and their role in cell signaling and regulation. *Front. Biosci.* 13, 6580–6603 (2008)
11. Lopez, R., Silventoinen, V., Robinson, S., Kibria, A., Gish, W.: Wu-blast2 server at the european bioinformatics institute. *Nucl. Acids Res.* 31, 3795–3798 (2003)
12. Zamboulis, L., Fan, H., Belhajjame, K., Siepen, J., Jones, A., Martin, N., Poulou-vassilis, A., Hubbard, S., Embury, S., Paton, N.: Data access and integration in the iSpider proteomics grid. In: Leser, U., Naumann, F., Eckman, B. (eds.) *DILS 2006. LNCS (LNBI)*, vol. 4075, pp. 3–18. Springer, Heidelberg (2006)
13. Labarga, A., Valentin, F., Anderson, M., Lopez, R.: Web services at the european bioinformatics institute. *Nucl. Acids Res.* 35 (July 2007)
14. Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M.R., Wipat, A., Li, P.: Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics* 20, 3045–3054 (2004)
15. Jenkinson, A., Albrecht, M., Birney, E., Blankenburg, H., Down, T., Finn, R., Hermjakob, H., Hubbard, T., Jimenez, R., Jones, P., Kahari, A., Kulesha, E., Macias, J., Reeves, G., Prlic, A.: Integrating biological data - the distributed annotation system. *BMC Bioinformatics* 9(suppl. 8) (2008)
16. Rice, P., Longden, I., Bleasby, A.: Emboss: The european molecular biology open software suite. *Trends Genet.* 16, 276–277 (2000)
17. Cote, R., Jones, P., Martens, L., Kerrien, S., Reisinger, F., Lin, Q., Leinonen, R., Apweiler, R., Hermjakob, H.: The Protein Identifier Cross-Reference (PICR) service: reconciling protein identifiers across multiple source databases. *BMC Bioinformatics* 8(1) (2007)
18. Jimenez, R.C.C., Quinn, A.F.F., Garcia, A., Labarga, A., O'Neill, K., Martinez, F., Salazar, G.A.A., Hermjakob, H.: Dasty2, an Ajax protein DAS client. *Bioinformatics* (August 2008)

19. Orchard, S., Montechi-Palazzi, L., Deutsch, E.W., Binz, P.A., Jones, A.R., Paton, N., Pizarro, A., Creasy, D.M., Wojcik, J., Hermjakob, H.: Five years of progress in the standardization of proteomics data. In: 4th annual spring workshop of the HUPO-Proteomics Standards Initiative. Ecole Nationale Supérieure (ENS). Proteomics, Lyon, France, April 23-25, vol. 7, pp. 3436-3440 (October 2007)
20. Côté, R.G.G., Jones, P., Martens, L., Apweiler, R., Hermjakob, H.: The ontology lookup service: more data and better tools for controlled vocabulary queries. *Nucl. Acids Res.* (May 2008)
21. Bendtsen, J.D., Nielsen, H., von Heijne, G., Brunak, S.: Improved prediction of signal peptides: Signalp 3.0. *J. Mol. Biol.* 340, 783-795 (2004)
22. Julenius, K., Mølgaard, A., Gupta, R., Brunak, S.: Prediction, conservation analysis, and structural characterization of mammalian mucin-type o-glycosylation sites. *Glycobiology* 15, 153-164 (2005)
23. Gnad, F., Ren, S., Cox, J., Olsen, J., Macek, B., Oroshi, M., Mann, M.: Phosida (phosphorylation site database): management, structural and evolutionary investigation, and prediction of phosphosites. *Genome Biology* 8(11) (2007)
24. Juretić, D., Zoranić, L., Zucić, D.: Basic charge clusters and predictions of membrane protein topology. *J. Chem. Inf. Comput. Sci.* 42(3), 620-632 (2002)
25. Käll, L., Krogh, A., Sonnhammer, E.L.: Advantages of combined transmembrane topology and signal peptide prediction—the phobius web server. *Nucl. Acids. Res.* 35 (July 2007)

Design and Architecture of Web Services for Simulation of Biochemical Systems

Joseph O. Dada¹ and Pedro Mendes^{1,2}

¹ School of Computer Science and Manchester Centre for Integrative Systems Biology,
The University of Manchester, Manchester Interdisciplinary Biocentre,
131 Princess Street, Manchester M1 7DN, UK

² Virginia Bioinformatics Institute, Virginia Tech, Washington Street 0477,
Blacksburg, VA 24060, USA
{joseph.dada,pedro.mendes}@manchester.ac.uk

Abstract. Computer simulation of biochemical networks plays a central role in systems biology. While there are several software packages for modeling and simulation of these networks, they are based on graphical user interfaces that operate on the local computer. However, it is now often desirable to operate simulation tasks through a distributed computing framework where data can be gathered from different sources and distinct subtasks operated in different physical computers. In this paper we describe a web services implementation for the COPASI biochemical network simulator (CopasiWS). COPASI provides a range of numerical methods for simulation, optimization and analysis of biochemical reaction networks. Our aim is to allow easy integration of these powerful functionalities with local and remote services to provide a distributed computing platform for the simulation and analysis of biochemical models. One immediate result of this work is that simulation tasks are now available to be used in a platform- and language-independent manner as components of distributed workflows, for example using the Taverna workflow engine. We describe the CopasiWS architecture, key design and implementation issues, and illustrate the range of services available through a web portal interface (CopasiWeb).

1 Introduction

Systems biology is a new approach to biological research where experimental design and analysis are carried out based on a view of cells as systems of interacting components, rather than each of its molecular components in isolation. A major part of this new approach is fulfilled by mathematical and computer models of the underlying interaction network which are used for simulation. There are several software packages for modeling and simulation of these networks, such as COPASI [1], or CellDesigner [2], which are mostly based on graphical user interfaces that operate on the local computer. The field of biochemical simulation has advanced mostly in terms of the analytic frameworks and numerical algorithms, but not so much in terms of architecture, which is either based

on single-computer or client-server solutions. Recent advances in grid and distributed computing suggest that it would be desirable to operate modeling and simulation of biochemical systems in a language independent manner and in a distributed computing environment. Such a system would allow accessing data from different sources and carrying out different computational operations in different servers on the network resulting in distributed simulations.

An earlier solution that allows for interoperable modeling and simulation tasks is the Systems Biology Workbench (SBW) [3]. SBW is based on a communication bus and a simple messaging system that allows applications to interoperate; SBW contains several applications that carry out specialized operations, from computational analysis to visualization. Because the SBW messaging bus is based on sockets, it is possible to run tasks in different computers and thus this system already allowed for distributed computing. However, SBW is based on a non-standard messaging protocol which is adopted only by a few specialized applications. Unfortunately this means that non-systems biology applications are not able to link with the SBW messaging system and thus this pioneering effort is unlikely to fulfill the full potential of distributed computing.

Web services are a widely used standard for interoperable machine-to-machine interaction over a network. Web services follow standard protocols approved by the W3C organization. A Web service is a system implemented by a software agent capable of sending and receiving messages defined through the Web services description language (WSDL, an XML-based language), and passed through the HTTP protocol. WSDL makes possible the discovery of web services through service brokers.

The advantage of using Web services to facilitate distributed computing is that unlike the case with SBW many more applications are immediately available, even outside the realm of systems biology. For example most bioinformatic resources are accessible through Web services (e.g. the KEGG [4], NCBI [5] and EBI databases [6], as well as sequence analysis tools). There are also many other scientific resources and even commercial services (e.g. Amazon.com). More specific to systems biology, the Biomodels database [7], MIRIAM [8], and the Systems Biology Ontology [9] are accessible through Web services. Workflow Management Systems such as Taverna [10] and Triana [11] allow programs to be created that combine these Web services in arbitrarily complex workflows – this allows the output of an application to become the input to another, resulting in computations carried out across the network, each task by specialized providers.

The aim of this paper is to describe a Web services implementation of the COPASI biochemical network simulator (hereafter named CopasiWS). COPASI [1] provides a range of numerical methods for simulation, optimization and analysis of biochemical reaction networks. COPASI, which is the successor to Gepasi [12], is available for all common operating system platforms and is currently supplied in two versions: one based on a graphical user interface (CopasiUI) and one that is entirely driven through the command line (CopasiSE). CopasiSE was designed such that simulations can be processed in batch mode without user intervention. COPASI is able to read the SBML standard format for systems biology network

models [13], but also has its own file format (CopasiML, also based on XML). While SBML only encodes a model, CopasiML also encodes all of the operations to be carried out with the model and definitions of report files containing output from the simulations.

Providing a Web services interface to COPASI will allow usage of simulation and modeling tasks in distributed workflows, and it makes possible the creation of simulation servers where computing cycles could be offered. Because COPASI carries out a wide range of analyses the result of this work is a large set of specific modeling and simulation Web services. The rest of this paper describes the design, architecture, and prototype implementation of CopasiWS. It ends with an illustration of the range of available services using a simple Web portal interface. Finally a discussion is presented of the perspectives that Web services can bring to systems biology.

2 Design of COPASI Modeling and Simulation Web Services (CopasiWS)

Software clients interact with Web services through interfaces to use the functionality provided by the service. In order to fulfill the client request, the Web service interacts with the appropriate business logic, which may also need to communicate with the application resource to accomplish the client request. The flow of information to accomplish a client request in CopasiWS can be modeled abstractly as a three-layer architecture as shown in Fig. 1. The design and implementation of CopasiWS need to take into consideration the various components that will reside in each of the layers. In this section, we describe the issues that had to be considered in the design of CopasiWS and details of these layers.

2.1 CopasiWS Design Issues

Web Service Interface Development Approach. A top-down approach was used to develop the interface to avoid interoperability problems and to conform with WS-I [14]. Basically the development starts with the definition of the WSDL. A stub code for the service is generated automatically and the developer then completes the implementation of the Web service.

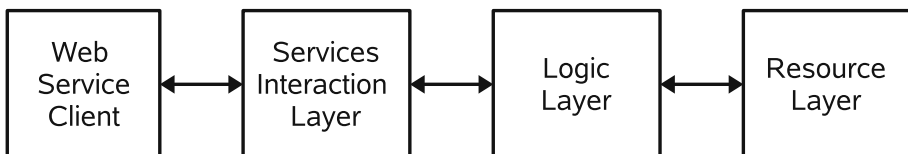


Fig. 1. A three-layer architecture model of CopasiWS. It shows a typical flow of information to accomplish client request to CopasiWS.

Service Granularity. One of the main issues in building Web services is the identification of the right granularity for the services provided. The COPASI user interface defines independent tasks and thus it was natural to follow this granularity for CopasiWS. Each COPASI task has specific input parameters and generates output in predefined text files (many are tab-delimited) or alternatively in some user-defined format. Some tasks have quite simple interfaces, while others are complex and require many parameters to be defined. For example, the time course simulation task can be carried out with deterministic, stochastic or hybrid algorithms, each one requiring a set of specific control parameters. An alternative to providing each task as a single service would be to process the entire specifications of a CopasiML document, like what is done by the command-line batch tool CopasiSE. Providing each of the tasks as a single Web service facilitates composing high-level workflows on top of these basic services. This has the further advantage that since each service only runs one task, it becomes natural to pass the model as an SBML model because then the remaining inputs are only a few other parameters. This is advantageous because otherwise client software would require being able to understand the CopasiML syntax and semantics (CopasiML is specific for COPASI while SBML is understood by numerous packages and a library is available [15] to manipulate files encoded in SBML).

Synchronous versus Asynchronous Web Services. Some COPASI tasks can take very short runs (seconds or less), however others can be very long (hours or more). Task execution time is usually proportional to the size of a model, but even for a single model different tasks are much slower than others (e.g. a single steady state calculation takes a lot less time than a fine-grained scan of parameter space). Usually tasks that take short computing times are better operated in a synchronous mode, where the client blocks waiting for the final results from the service. Others that are long are better operated in an asynchronous mode where the client first requests the service then checks at discrete times whether the service has ended before retrieving results. Unfortunately it is impossible to predict the length of a task and thus it is not possible to select synchronous vs. asynchronous modes automatically; the requestor of the service must decide a priori which mode of operation to use. To address these two modes of operation each of the tasks are implemented by one synchronous and one asynchronous Web service.

The synchronous implementation is straightforward since it consists of simply creating an intermediate CopasiML input file, running CopasiSE and returning the results in an appropriate format. On the other hand, asynchronous services could be implemented through various patterns of operation. We choose request/reply operations with a polling pattern that consist of many synchronous operations in which the clients initiate all the interactions (i.e. no call-back functions are needed). In this approach, the client is issued an identifier in the first call to the service, and this is then used for subsequent operation calls. This approach decouples the client and service provider and make the implementation of client applications very simple.

Parameter Types of Web Service Operations. A client calls an appropriate method with parameters on the service interface to use the functionality of the service. The parameter types determine the style of the interface. Parameters may be passed as either JAX-RPC value types or XML documents. CopasiWS uses the XML document mode of parameter passing because some of the COPASI tasks require a complex set of parameters.

COPASI Task Input. In order to maximize the level of compatibility of CopasiWS with other systems biology software tools, it is important that it can take SBML files as input. This means that a series of additional parameter values must be passed as input, since they are needed for the operation of the task but are not part of the SBML specification. Alternatively, the CopasiML format contains all required parameters and therefore we also defined versions of the services that take this format as input. The latter mode suggests that the CopasiUI GUI-based software itself could be a client of CopasiWS; this would allow users to choose between running tasks locally or remotely. The SBML input versions of the services are better suited for workflows that use other SBML software, such as the Biomedb database or the a wrapper to libSBML [28]. The CopasiWS suite also includes a service to convert between the SBML and CopasiML formats. When clients use the SBML version of services, the system translates the SBML into CopasiML, then sets the appropriate task and associated parameters in the CopasiML document and then calls CopasiSE to run the task.

COPASI Task Output. COPASI results can be formatted in flexible ways, but there are already some predefined tab-delimited data formats for specific tasks. These, however, are not very useful for Web services and make further processing of the output data by the clients difficult. It is usual to have Web services results be encoded in XML so we decided to format task results in the Systems Biology Results Markup Language (SBRML), which is an XML-based language to encode systems biology simulation results and experimental data [16].

Many modeling tasks are oriented towards changing details of a model rather than producing numerical results. The results of these tasks are then better suited to be expressed directly in SBML or CopasiML. The Web services that encapsulate those tasks then produce output in exactly the same format as their input. An example of such tasks is parameter estimation, which takes a model together with some data and changes some numeric parameters of the model to best match the data; the result being a new version of the model.

Experimental data. COPASI's parameter estimation task has a more complex input than other tasks as it requires not only a model but also target data that the model should fit. COPASI takes these data in a tab-delimited format where columns of data correspond to some model entities. The data columns must then be mapped to model entities, a tiresome task that is usually done through the GUI by the user. However the purpose of the SBRML format mentioned above [16] is exactly to encapsulate data that is mapped to a model and so it is the most appropriate means to pass these data to the parameter estimation Web

service. The current implementation of this Web service decodes the SBRML and converts it to COPASI's native format; later it is expected that COPASI itself be able to read SBRML directly, which will simplify the present code.

Simulation Resource Management. To process long simulation tasks CopasiWS provides asynchronous processing. For each asynchronous simulation request the system creates a simulation resource. A simulation resource is identified by a unique identifier and has associated a number of files which are stored transiently in the server (the SBML or CopasiML files, simulation output file, experimental data file, resource life time, etc.) CopasiWS therefore needed a mechanisms to manage the resources created by the users.

Simulation Engine Management. CopasiSE, the command line driven version of COPASI, is the ultimate executor of the actual tasks and resides in the resource layer (see Fig. 1). Because CopasiWS is available to many potential clients simultaneously, there will be at times the need to access several instances of CopasiSE simultaneously. This calls for having high-performance or high-throughput computing resources in the server, and this also requires specific modules to manage these resources. As an example we have implemented a high-throughput simulation engine module using the University of Wisconsin's Condor system [17]. Condor creates pools of computers which make their computational power available when idle. In our prototype, CopasiSE jobs are queued to a Condor pool whenever needed by asynchronous Web service calls. Of course, for a responsive and fast service one should set aside sufficient dedicated machines to deal with the required throughput (though these could also be managed through the Condor system if needed).

2.2 Design of CopasiWS Interface

Most of the efforts in designing a Web service interface are spent on the service operations. In this case there was a clear guideline consisting in the COPASI user interface since the Web services follow the same division of tasks. Here we briefly describe the functionalities of each task.

Importing/Exporting and Validation Tasks. COPASI provides methods for converting files from SBML to CopasiML format and vice versa, and also to validate files in either of these formats. These tasks are exposed as Model Processor Web service with operations for converting between SBML and CopasiML, and also for XML schema validation.

Steady State Task. Systems biology models are dynamic models which may be able to reach steady states (*i.e.* where the values of their variables do not change). This task finds one steady state of the dynamical system if it exists. The numerical method used in this task has a number of control parameters, including a tolerance value, and which numerical methods to solve for the steady state (Newton-Raphson and/or numerical integration).

Time Course Task. Since these models are dynamical systems, one of the most basic tasks is to determine a trajectory of the system given an initial condition (which is specified in the input files). COPASI provides different methods for calculating the trajectory: *Deterministic* (which uses the LSODA ordinary differential equation solver [18]), *Stochastic* (using Gillespie's stochastic simulation algorithm [19]) and *Hybrid simulation* (using a combination of the previous two [1]). Each of these methods has a different set of control parameters. All of these tasks are available through a single Time Course Web service (which can use any one of the three methods).

Metabolic Control Analysis (MCA) Task. MCA is a special type of sensitivity analysis which has a particular interpretation for metabolic networks [20] and quantifies how much the rates of reactions affect the concentrations or fluxes at the steady state. This task has only one control parameter that dictates whether the coefficients are to be scaled or unscaled.

Optimization Task. COPASI provides a framework to find optima of any model state variable or any arbitrary function of state variables. It does this by providing a series of alternative numerical optimization methods, from traditional gradient-based to stochastic global optimizers. Each of these algorithms requires its own particular control parameters. The interface of the Optimization Web service allows the calling function to choose the optimization method and the objective function.

Parameter Estimation Task. Biochemical models depend on many numerical parameters, but quite frequently their values are unknown and have to be estimated from some data set [21]. This task makes use of the optimization algorithms mentioned in the previous section to minimize a least squares function (representing the distance between the model simulation and a set of experimental data). Because this task usually requires high computational demands, we have implemented it as an asynchronous Web service only. The client of this service needs to first call the *CreateSimulationResource* operation, which creates a simulation resource with a unique id which is returned to the client. The client then uses this resource id as one of the parameters for the subsequent operation calls. There are operations for submitting the model, for submitting the experimental data, for starting the simulator and for checking and collecting the results. The result of this Web service is a model in SBML or CopasiML formats, plus a set of statistics of the goodness of fit.

2.3 CopasiWS WSDL Development

We used the functionalities of the COPASI tasks described above to develop a WSDL specification for CopasiWS. Operations are grouped into port types, which describe abstract end points of each Web service. The message element of WSDL defines the data elements of an operation. The message data types are defined using a platform neutral XML Schema syntax. Because there are some data types that are common to some of the Web services, we created a separate

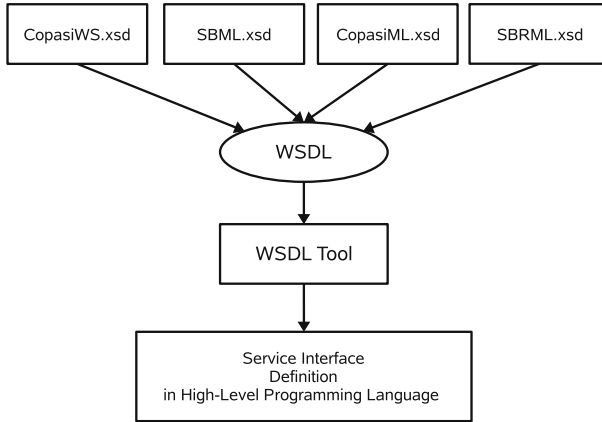


Fig. 2. CopasiWS WSDL development process

XML Schema for the data types (CopasiWS.xsd). We use these data types together with other XML Schema data types for biochemical models (SBML.xsd and CopasiML.xsd) and simulation results (SBRML.xsd) to develop the WSDL for the Web services. Appropriate WSDL toolkit was used to generate the service interface in a specific programming language as represented in Fig. 2.

3 CopasiWS Architecture

Figure 3 depicts the overall architecture of CopasiWS. The three layers already depicted in Fig. 1 are shown here with their internal components. The client communicates with the Web services interaction layer using standard Web service calls, while the interaction layer communicates with the logic layer using a local proprietary interface mechanisms. The logic layer again communicates with the resource layer in order to accomplish the client requests. This model provides a loose coupling between layers and makes it easy to implement components of each layer independently. It also allows for changes in the internal layers while keeping the same interface to the outside world.

3.1 Resource Layer

This layer contains the simulation engine (CopasiSE) that runs the simulation tasks submitted by the logic layer. It also contains a database server that holds relevant information about users and the simulation results.

3.2 Logic Layer

This layer contains various components that help the services interaction layer to accomplish the client request. It contains the following components: simulation

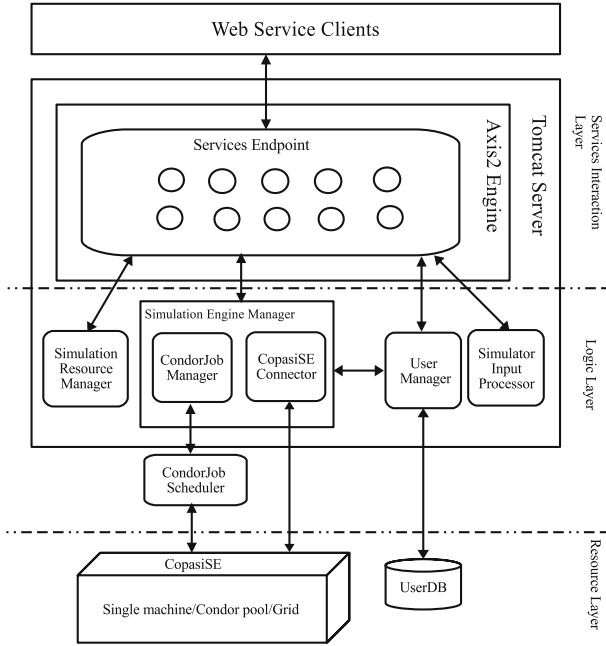


Fig. 3. CopasiWS Architecture

resource manager, user manager, simulator input processor, simulation engine manager and the Condor job scheduler [17], which is a third party component (other grid queue managers could be implemented here as alternatives). The components have well-defined interfaces for communicating with the services interaction layer.

3.3 Services Interaction Layer

This layer provides the glue between the clients and the service functionalities. All the Web service interfaces reside in this layer, and it is responsible for starting the processes in the logic layer in response to client requests. It is also responsible for routing requests to the appropriate components of the logic layer.

The communication between a service endpoint in the interaction layer and the components of the logic layer is determined by the type of request from the clients. An example of how the components of the layers send messages to one another to accomplish a client request (e.g. *runSimulator* operation call to a service endpoint) is shown in Fig. 4.

4 Prototype Implementation, Testing and Example Usage

We have implemented a prototype of the CopasiWS. The services interaction layer is developed in Java programming language [22]. We used an Apache Axis 2

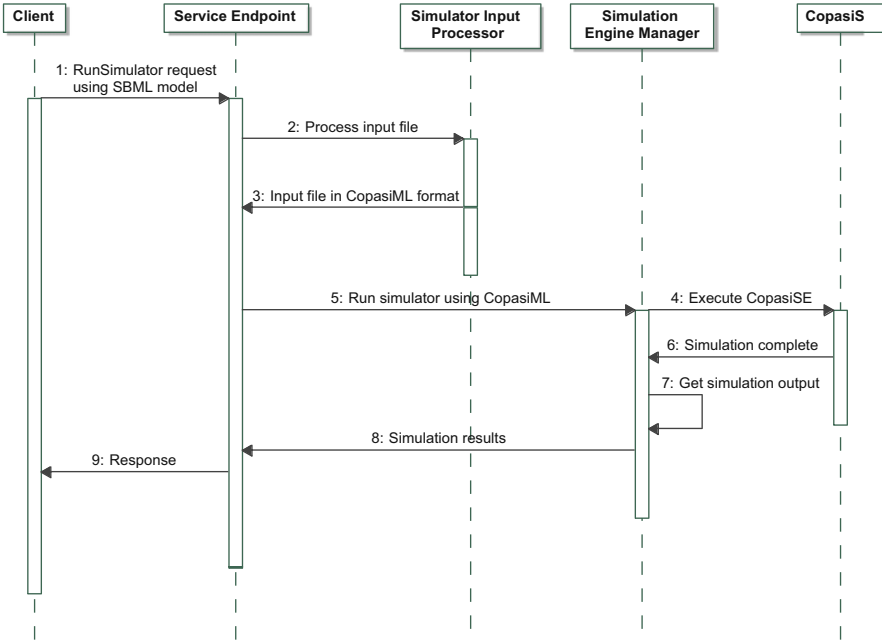


Fig. 4. Sequence of messages between components of layers in CopasiWS

toolkit [23] to generate the interface definition of the services in Java from their WSDL as discussed in section 2 and then added each service interface implementation code to complete the coding process. The services run in an Apache Axis 2 engine, which is hosted on an Apache Tomcat server [24]. All the components in the logic layer excluding the scheduler (the third party component) are also developed in Java.

The CopasiSE that resides in the resource layer is the command line version of COPASI (originally written in C++, though only the executable binary is used here) and runs on the actual server; alternatively it can run on a Condor pool or on a grid system. The user database is presently implemented using a file system. Future versions of CopasiWS will likely use a database management system for the user database.

Interested users should consult [25] to obtain the WSDL for simulation services presently available in CopasiWS. Currently there is no requirement for users to register before using the CopasiWS synchronous version. However those interested in using the asynchronous version should contact the authors to obtain username and password.

4.1 Web Portal User Interface (CopasiWeb)

To test the range of services available in CopasiWS, we developed a web portal user interface (CopasiWeb). CopasiWeb is one of the possible clients of

CopasiWS. It is basically divided into two parts: the first part is the simple HTML page that appears in user's browser and provides a means of interacting with the CopasiWS, while the second part is the Web service manager, which is hosted on a Tomcat Server. This converts the user's request from the browser into Web service calls that are directed to the CopasiWS. CopasiWeb is based on the Model View Controller (MVC) architecture and is implemented using Apache Struts 2 [26]. A detailed description of CopasiWeb is outside the scope of this paper.

CopasiWeb is available from [27]. There is no requirement for users to register to use it, except if they want to execute the asynchronous services as described above.

4.2 Example Usage

CopasiWS provides a suite of services that can easily be combined with other applications and services to provide a flexible platform for modeling, simulation and analysis of biological processes. The steps involved in running CopasiWS depend on which version (synchronous or asynchronous) a client wants to use. For the synchronous version, the communication with the service follows a simple request and response approach. The client sends a run simulator request to the service and waits for a response. In the case of asynchronous version, client needs to follow a sequence of steps to execute the service. Here we use parameter estimation Web service to illustrate how to use the asynchronous version.

The parameter estimation task/service as earlier described is used to estimate the values of the unknown parameters in biochemical models. To carry out model parameter estimation process, the user needs to make the following available to the service:

1. a biochemical model in SBML format;
2. experimental data in SBRML format;
3. information about the model parameters to estimate (i.e. model parameter identifier with lower and upper boundary values);
4. optimization method and its parameter values.

The above data are then passed to the service by the client application through the following sequence of steps using appropriate service operations:

1. Client creates a simulation resource using username and gets resource ID in return;
2. Client uses the resource ID to submit biochemical model in SBML;
3. Client submits experimental data in SBRML format using the resource ID;
4. Client sets the model parameter items to estimate and optimization method to use using the resource ID;
5. Client starts the simulator using the resource ID;
6. Client periodically checks the simulation status (polling approach) using the resource ID;
7. Client gets the simulation results (usually updated SBML model);
8. Client destroys the created resource. A resource that is not destroyed by the client will be destroyed by the system after its lifetime.

This sequence of steps can easily be automated using a workflow management system. For instance a systems biology workflow can be constructed to retrieve the SBML model from the Biomodels database [7] and experimental data in SBRML from other Web services and then use these as input to the parameter estimation service to estimate the unknown SBML model parameters. Each of these tasks would be operating from a different server on the Internet.

5 Conclusion and Further Work

Systems biology is an increasingly popular mode of research in the biological sciences which makes heavy use of computational methods and resources. We have constructed a collection of Web services, named CopasiWS, that expose the functionalities of the systems biology modeling and simulation software COPASI. To our knowledge this is the first implementation of systems biology simulation tasks conforming to Web services standards. The availability of such methods through the means of Web services will allow a more flexible approach to computational systems biology where data and services are distributed throughout the Internet. An obvious use of these Web services would be to provide the simulation tasks that are currently only available through a graphical user interface in a client-server model where heavy computational resources are hosted behind this interface. Another new computing paradigm that these Web services allow, is the construction of distributed workflows, for example using Taverna [10], which is already widely used in the related field of bioinformatics. Furthermore this will also open up the possibility of running complex simulations across the network. For example this interface could easily be exploited to construct a multi-scale simulation environment where the COPASI tasks fulfill one of the levels (e.g. cellular) while other Web services, or Web services clients, implement other levels (e.g. tissue or organ).

Future work will consist of implementing access security features and further user management functions. Because it is likely that computing power needs will increase, we plan on implementing methods to access Grid resources in the logic layer. Additionally, we would like to demonstrate the power of combining Web services by creating distributed workflows that access other resources, such as the Biomodels database.

Acknowledgments. We thank Stefan Hoops, Douglas Kell, Peter Li, Norman Paton, Irena Spasić and Neil Swainston for many helpful discussions. We greatly benefited from the lessons taught by Anwar Ul Haq's first prototype of COPASI Web services. We thank the generous financial support by the BBSRC and EPSRC to this project through funding of the Manchester Centre for Integrative Systems Biology. The MCISB is a Centre of the Biotechnology and Biological Sciences Research Council.

References

- [1] Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., Kummer, U.: COPASI — a COMplex PATHway SIMulator. *Bioinformatics* 22, 3067–3074 (2006)
- [2] Funahashi, A., Tanimura, N., Morohashi, M., Kitano, H.: CellDesigner: a process diagram editor for gene-regulatory and biochemical networks. *Biosilico* 1, 159–162 (2003)
- [3] Sauro, H.M., Hucka, M., Finney, A., Wellock, C., Bolouri, H., Doyle, J., Kitano, H.: Next Generation Simulation Tools: The Systems Biology Workbench and BioSPICE Integration. *A Journal of Integrative Biology* 7(4), 355–372 (2003)
- [4] Ogata, H., Goto, S., Sato, K., Fujibuchi, W., Bono, H., Kanehisa, M.: KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research* 27(1), 29–34 (1999)
- [5] Baxevanis, A.D.: Searching NCBI databases using Entrez. *Current protocols in bioinformatics* 24, 1.3.1–1.3.26 (2008)
- [6] Labarga, A., Valentin, F., Anderson, M., Lopez, R.: Web services at the European bioinformatics institute. *Nucleic Acids Research* 35, W6–W11 (2007)
- [7] Le Novère, N., Bornstein, B., Broicher, A., Courtot, M., Donizell, M., Dharuri, H., Li, L., Sauro, H., Schilstra, M., Shapiro, B., Snoep, J., Hucka, M.: Biomodels database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Research* 34, D689–D691 (2006)
- [8] Le Novère, N., Finney, A., Hucka, M., Bhalla, U.S., Campagne, F., Collado-Vides, J., Crampin, E.J., Halstead, M., Klipp, E., Mendes, P., Nielsen, P., Sauro, H., Shapiro, B., Snoep, J., Spence, H., Wanner, B.: Minimum Information Requested In the Annotation of Biochemical Models (MIRIAM). *Nature Biotechnology* 23(12), 1509–1515 (2005)
- [9] Systems Biology Ontology (SBO), <http://www.ebi.ac.uk/sbo/>
- [10] Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M.R., Li, P., Oinn, T.: Taverna: a tool for building and running workflows of services. *Nucleic Acids Research*, W729–W732 (2006)
- [11] Taylor, I., Shields, M., Wang, I., Harrison, A.: The Triana Workflow Environment: Architecture and Application. In: Taylor, I.J., Deelman, E., Gannon, D.B., Shields, M. (eds.) *Workflows for e-Science*. Scientific Workflows for Grids, pp. 320–339. Springer, London (2007)
- [12] Mendes, P.: GEPASI: A software package for modeling the dynamics, steady states and control of biochemical biology and other systems. *Computer Application in the Biosciences* 9(5), 563–571 (1993)
- [13] Hucka, M., Finney, A., Sauro, H.M., Bolouri, H., Doyle, J.C., Kitano, H., Arkin, A.P., Bornstein, B.J., Bray, D., Cornish-Bowden, A., Cuellar, A.A., Dronov, S., Gilles, E.D., Ginkel, M., Gor, V., Goryanin, I., Hedley, W.J., Hodgman, T.C., Hofmeyr, J.H., Hunter, P.J., Juty, N.S., Kasberger, J.L., Kremling, A., Kummer, U., Le Novère, N., Loew, L.M., Lucio, D., Mendes, P., Minch, E., Mjolsness, E.D., Nakayama, Y., Nelson, M.R., Nielsen, P.F., Sakurada, T., Schaff, J.C., Shapiro, B.E., Shimizu, T.S., Spence, H.D., Stelling, J., Takahashi, K., Tomita, M., Wagner, J., Wang, J.: The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics* 19, 524–531 (2003)

- [14] Web Services Interoperability Basic Profile, 1.1, <http://www.ws-i.org/Profiles/BasicProfile-1.1.html>
- [15] Bornstein, B.J., Keating, S.M., Jouraku, A., Hucka, M.: LibSBML: An API Library for SBML. *Bioinformatics* 24(6), 880–881 (2008)
- [16] Dada, J.O., Paton, N.W., Mendes, P.: Systems Biology Results Markup Language — Structure and Facilities for Systems Biology Results Representation (SBRML Specification) (2008), <http://www.comp-sys-bio.org/tiki-index.php?page=SBRML>
- [17] Thain, D., Tannenbaum, T., Livny, M.: Distributed Computing in Practice: The Condor Experience. *Concurrency and Computation: Practice and Experience* 17, 2–4 (2005)
- [18] Petzold, L.: Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *SIAM Journal on Scientific and Statistical Computing* 4(1), 136–148 (1983)
- [19] Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry* 81(25), 2340–2361 (1977)
- [20] Fell, D.: *Understanding the Control of Metabolism*. Portland Press, London (1996)
- [21] Mendes, P., Kell, D.: Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation. *Bioinformatics* 14(10), 869–883 (1998)
- [22] Java programming language, <http://java.sun.com>
- [23] Axis toolkit site, <http://ws.apache.org/axis/>
- [24] Tomcat Apache servlet site, <http://tomcat.apache.org/>
- [25] Copasi Web Services, <http://turing.mib.man.ac.uk:8080/CopasiWS/services/listServices>
- [26] Apache Struts 2, <http://struts.apache.org/2.x/>
- [27] CopasiWeb: Web Portal Interface to CopasiWS, <http://turing.mib.man.ac.uk:8080/CopasiWeb/CopasiWebUI/>
- [28] Li, P., Oinn, T., Soiland, S., Kell, D.B.: Automated manipulation of systems biology models using libSBML within Taverna workflows. *Bioinformatics* 24(2), 287–289 (2008)

An Integration and Analysis Pipeline for Systems Biology in Crop Plant Metabolism

Stephan Weise, Christian Colmsee, Eva Grafahrend-Belau, Björn Junker, Christian Klukas, Matthias Lange, Uwe Scholz, and Falk Schreiber

Leibniz Institute of Plant Genetics and Crop Plant Research (IPK),
Corrensstr. 3, 06466 Gatersleben, Germany
{weise,colmsee,grafahr,junker,klukas,lange,
scholz,schreibe}@ipk-gatersleben.de

Abstract. To advance the comprehension of complex biological processes occurring in crop plants (e.g. for improvement of growth or yield) it is of high interest to reconstruct and analyse detailed metabolic models. Therefore, we established a pipeline combining software tools for (1) storage of metabolic pathway data and reconstruction of crop plant metabolic models, (2) simulation and analysis of stoichiometric and kinetic models and (3) visualisation of data generated with these models. The applicability of the approach is demonstrated by a case study of cereal seed metabolism.

1 Introduction

Crop plants form one of the main sources of human and animal nutrition and importantly contribute to chemical or pharmaceutical industry and renewable resources [1,2,3,4]. In order to achieve an improvement of growth and yield of crop plants it is necessary to understand biological processes on a detailed level [5,6].

Due to the increasing amount of data obtained by modern high-throughput technologies it becomes more difficult to perform all necessary experiments conventionally. However, such large amounts of data enable new analysis and modelling techniques. In order to cope with these challenges, systems biology aims to understand biological processes as a whole and to map onto mathematical models [7], thus enabling *in silico* experiments.

Mathematical modelling of metabolism enables analysis of the structure, dynamics and behaviour of metabolic networks. With the help of these models, understanding of complex processes can be verified and extended, new hypotheses can be generated, and suitable targets for metabolic engineering can be identified by exploring *in silico* scenarios. In plant metabolism, different methods for mathematical modelling are constantly gaining attention [8,9,10].

To deal with such models, several aspects have to be taken into consideration: model representation, model exchange, model analysis and model visualisation.

Data about biological processes in plants is available from various, often ambiguous, sources and thus needs to be integrated and persistently stored in a central, well-structured repository. To ensure high data quality, the integrated

data needs to be curated manually [11]. This comprises, for example, to distinguish different growth and developmental stages using controlled vocabulary. Once the data is prepared this way it can be used together with different tools for visualisation, simulation and analysis purposes. Therefore, standardised exchange formats such as the Systems Biology Markup Language (SBML) [12] or the Biological Pathways Exchange Language (BioPAX) [13] should be used.

Depending on the objective of the modelling process and the available data used for model reconstruction, different model analysis techniques can be applied to the reconstructed metabolic model ranging from purely stoichiometric to kinetic approaches of model analysis. With each of these modelling techniques being characterised by certain advantages and drawbacks, an analysis pipeline offering a variety of different model analysis techniques is needed to support user-friendly and flexible model simulation and analysis.

With increasing amount of large-scale experimental data, simulation results and large biological networks, visualisation methods become more and more important for data analysis. Advanced visualisation methods aim at bringing the data in a form that, on the one hand gives an overview about the overall system, and on the other hand provides sufficient detail. Static visualisation is inadequate for large scale data exploration, thus in order to assist modern biological research dynamic visualisation and user-friendly interaction methods need to be implemented in supportive tools.

This paper describes a pipeline for supporting the research on crop plant metabolic models. It comprises the following steps: (1) data management and individual, user-specific model reconstruction, (2) stoichiometric and kinetic model analysis and (3) model and flux visualisation. Each of the steps is described focusing on tools and methods developed. The applicability of the pipeline is shown by a case study of storage metabolism in developing seeds of the agriculturally important crop species barley.

2 Methods

For the integration and analysis of data describing metabolic networks of plants a pipeline was developed, which is summarised in Figure 1. The respective steps are described in detail below.

2.1 Model Storage and Reconstruction

The reconstruction of plant metabolic models is an important step for a better understanding of biological processes. Therefore, detailed metabolic information up to compartment level needs to be collected and managed in a well-structured way. On this account we developed MetaCrop (<http://metacrop.ipk-gatersleben.de>) [14], a manually curated repository of high quality data of seven major crop plants with high agronomical importance (*Hordeum vulgare* (barley), *Triticum aestivum* (wheat), *Oryza sativa* (rice), *Zea mays* (maize), *Beta vulgaris* (beet), *Solanum tuberosum* (potato) and *Brassica napus* (canola)) and two model plants widely

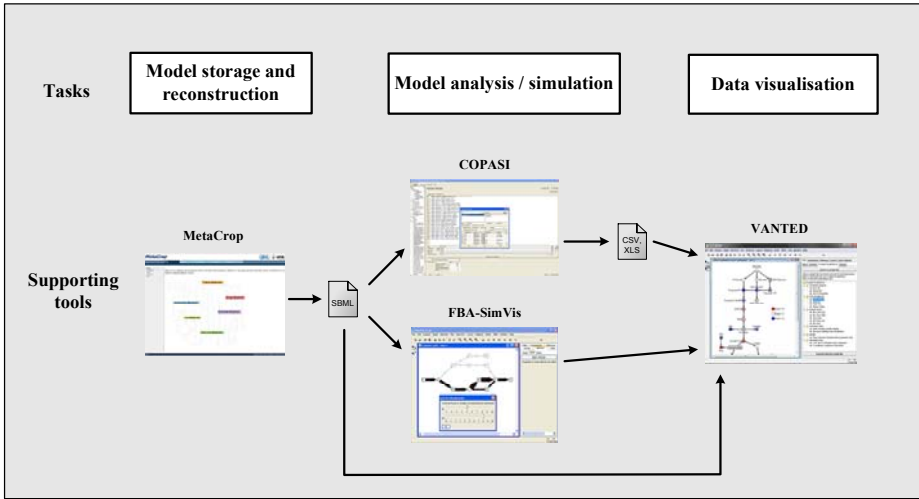


Fig. 1. Comprehensive pipeline of integration and analysis of pathway related data

used in plant research (*Arabidopsis thaliana* (thale cress) and *Medicago truncatula* (barrel medic)). MetaCrop is available for scientists working in the area of plant research, thus accelerating the process of data curation.

MetaCrop uses the Meta-All software [15]. Major concepts of MetaCrop are *substances* and *conversion processes*. The latter of which should be understood as a reaction or a translocation, which can be either actively or passively. Substances play certain roles within these conversion processes, such as substrate, product, catalyst or inhibitor. To enable the successive construction of metabolic models, conversions can be combined to pathways and pathways to super-pathways.

Fine-grained information can be stored with every conversion element managed in MetaCrop, such as reaction or translocation type, formula, synonyms, EC numbers, literature references and also kinetic data like V_{max} values, affinity or inhibitor constants. Moreover, all this information is assigned with the compartment the respective element is located in, thus considering that conversions take place at different locations inside an organism depending on the developmental state and environmental effects. Pathways can be stored as different parallel versions and furthermore, there is the possibility to assign quality tags due to different quality levels in the data used for MetaCrop.

Since there are only a few models for crop plants existing at all, the time-consuming process of manual data curation is indispensable. Data can be imported into MetaCrop using either the standardised SBML format [12] or a web interface and can then be curated manually. To enable visualisation, simulation and analysis metabolic pathway data can be exported using the SBML format. Therefore, MetaCrop offers an export functionality, which is integrated into the graphical user interface. It enables a user to compose an individual pathway model by adding elements such as reactions, translocation processes or even whole pathways into a ‘shopping cart’ step by step. By means of a wizard,

export settings can be defined then, e. g. the use of certain kinetics or a restriction to data from a certain species only. The generated SBML file contains function definitions, unit definitions, all involved species, annotations and the reactions including their kinetic data. Additionally, it is possible to export structural models only. Both SBML import and export can also be performed on command line thus enabling batch runs.

2.2 Model Analysis and Simulation

Stoichiometric Model Analysis. Stoichiometric model analysis or constraint-based modelling is based on the knowledge about the topological structure of the metabolic network under consideration. Due to the advantage of not requiring the knowledge of enzyme kinetic properties, constraint-based modelling approaches such as Flux balance analysis (FBA) have become an important approach for understanding the capabilities and properties of metabolic networks [16,17].

To provide a user-friendly environment for the constraint-based analysis of crop plant metabolic models, we developed FBA-SimVis (<http://fbasimvis.ipkgatersleben.de>), a VANTED ([18], see section 2.3) plug-in for integrated constraint-based model analysis and visualisation. The plug-in integrates various constraint-based analysis techniques (Flux balance analysis, Knock-out analysis, Robustness analysis and Flux variability analysis) with interactive and dynamic visualisation routines to support the quantitative analysis of stoichiometric models of plant metabolism. Due to the dynamic and visual exploration of simulated flux data resulting from model analysis, aimed at facilitating the analysis and interpretation of metabolic fluxes in response to genetic and/or environmental conditions, FBA-SimVis provides a comprehensive understanding of constraint-based metabolic models in both overview and detail.

To perform constraint-based analysis of a crop plant metabolic model exported from MetaCrop, the model has to be imported into FBA-SimVis as a SBML file, which forms the basis for subsequent stoichiometric model analysis and flux visualisation.

Kinetic Model Analysis. Kinetic models are the most detailed models of metabolism and allow most fine-grained predictions on metabolic behaviour. However, several issues arise due to their complexity: kinetic models require detailed kinetic knowledge about the modelled enzymes, and calculations quickly reach the limits of affordable computational power. Furthermore, the establishment of kinetic models is a tedious task requiring acquisition of various types of detailed information. In the integration and analysis pipeline presented in this paper, SBML files that were exported from MetaCrop contain all the necessary information to build a kinetic model and thus can be imported in various simulation software packages. As an example for a simulation software we use COPASI [19], which is a GUI-based tool that allows easy model editing and simulation also for non-experts. Once the model has been imported into COPASI, it can be edited and refined by the user, so that also data not contained in MetaCrop can be added. COPASI then allows to simulate the model as a

time-course, to calculate a steady-state and analyse its stability, to perform parameter estimation based on experimental data, to perform metabolic control analysis, etc. The calculated results are stored in text files which can be opened in spreadsheet programmes and thus passed on to data visualisation tools as explained in the next section. Furthermore, data can be exported in SBML format and can then be used to update MetaCrop.

2.3 Model Visualisation

VANTED (visualisation and analysis of networks containing experimental data) [18] is a platform-independent open source software which enables researchers to evaluate extensive experiment data from multiple *-omics* areas (transcriptomics, proteomics and metabolomics) and in the same way, the results of simulation studies. In order to support the analysis and visualisation of metabolic flux data, the data-mapping method, which initially was targeted at connecting experiment data to graph nodes, now supports the assignment of experimental datasets to graph edges.

Flexible network-integrated visualisation techniques may be used to visualise the data connected to graph nodes or edges. Available approaches include line, bar and pie charts as well as transformation functions, which map observed data to visual properties such as node size, edge width or graph element colours.

To support flexible transfer of graph models with various software tools and data sources, the following file formats are supported: GML, GraphML, DOT, Pajek .NET, KGML, SBML. Experiment data may be loaded into the system in the form of Excel spreadsheet files (XLS) or as comma separated value files (CSV).

3 Case Study

With the aim of getting a systemic understanding of cereal seed storage metabolism, a stoichiometric model of central metabolism in the developing endosperm of barley (*Hordeum vulgare*) was reconstructed by integrating biochemical, physiological and proteomic data derived from literature and databases into MetaCrop. The resulting compartmented stoichiometric model includes 257 conversion processes (193 reactions, 64 transport processes) and 234 metabolites, compartmentalised between the extracellular medium and the intracellular compartments cytosol, mitochondria and plastid.

To study grain yield and metabolic flux distributions under varying growth conditions, the model was subjected to Flux balance analysis using FBA-SimVis. Parameters necessary to perform FBA (e.g. maximum uptake and excretion rates) were extracted from published experimental results and the model was validated by comparing the simulation results to literature-based findings.

In general, the simulation results were found to be in good agreement with the main qualitative physiological characteristics of cereal seed storage metabolism. For example, the growth rate and the metabolic pathway pattern under fully

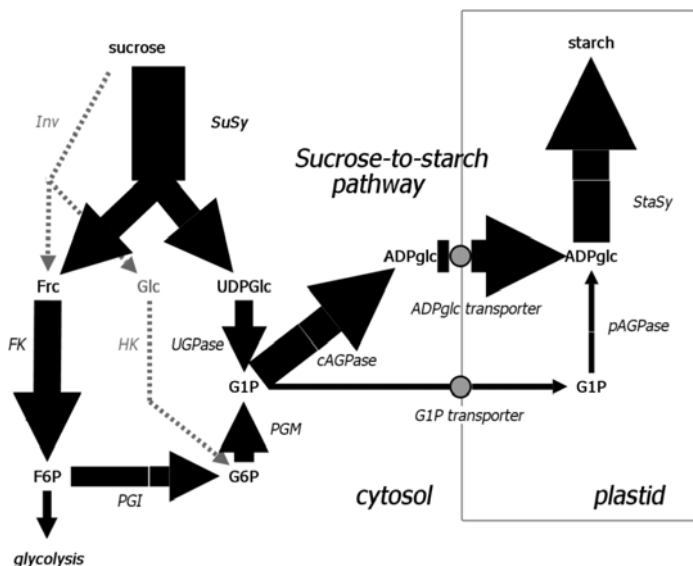


Fig. 2. Carbon flux map of the Sucrose-to-starch pathway predicted by the stoichiometric model of cereal seed metabolism under fully aerobic growth conditions (visualised with VANTED)

aerobic growth conditions predicted by the model were in accordance with published experimental results (see Figure 2 for an example). Thus, by providing an initial template for studying seed metabolic behaviour *in silico*, in future applications the model can be used to generate or test hypothesis on ways to improve grain and yield quality.

4 Conclusion

In this paper we presented a data integration and analysis pipeline for data about crop plant metabolism. It comprises the steps model storage and reconstruction, model analysis and model visualisation applying MetaCrop, FBA-SimVis / COPASI and VANTED, respectively. All tools used are publicly available. As a proof of concept a case study illustrating creation, analysis and visualisation of a model of cereal seed metabolism was shown. The pipeline as described here is applicable to all other crop plant species managed in MetaCrop. In principle, it could also be used for non-plant species by establishing a separate repository employing the Meta-All software.

Generally, the presented pipeline could be fully automated by applying workflow management systems, such as Taverna [20] or Kepler [21]. But on current data in crop plant biology we reckon the use of professionals' expertise as indispensable. Therefore, manual interaction is needed.

Acknowledgements

This work was partly supported by the German Federal Ministry of Education and Research (BMBF) under grants 031270-6A, 0315044A and 0315295.

References

1. Tilman, D., Hill, J., Lehman, C.: Carbon-Negative Biofuels from Low-Input High-Diversity Grassland Biomass. *Science* 314(5805), 1598–1600 (2006)
2. Metzger, J., Bornscheuer, U.: Lipids as renewable resources: current state of chemical and biotechnological conversion and diversification. *Applied Microbiology and Biotechnology* 71(1), 13–22 (2006)
3. Scheller, J., Guhrs, K.H., Grosse, F., Conrad, U.: Production of spider silk proteins in tobacco and potato. *Nature Biotechnology* 19(6), 573–577 (2001)
4. DellaPenna, D.: Nutritional Genomics: Manipulating Plant Micronutrients to Improve Human Health. *Science* 285(5426), 375–379 (1999)
5. Jenner, H.: Transgenesis and yield: what are our targets? *Trends in Biotechnology* 21(5), 190–192 (2003)
6. Carrari, F., Urbanczyk-Wochniak, E., Willmitzer, L., Fernie, A.: Engineering central metabolism in crop species: learning the system. *Metabolic Engineering* 5(3), 191–200 (2003)
7. Kitano, H.: Systems biology: A brief overview. *Science* 295(5560), 1662–1664 (2002)
8. Giersch, C.: Mathematical modelling of metabolism. *Current Opinion in Plant Biology* 3(3), 249–253 (2000)
9. Morgan, J., Rhodes, D.: Mathematical modeling of plant metabolic pathways. *Metabolic Engineering* 4(1), 80–89 (2002)
10. Poolman, M., Assmus, H., Fell, D.: Applications of metabolic modelling to plant metabolism. *Journal of Experimental Botany* 55(400), 1177–1186 (2004)
11. Zhang, P., Foerster, H., Tissier, C., Mueller, L., Paley, S., Karp, P., Rhee, S.: MetaCyc and AraCyc. *Metabolic Pathway Databases for Plant Research. Plant Physiology* 138(1), 27–37 (2005)
12. Hucka, M., Finney, A., Sauro, H.M., Bolouri, H., Doyle, J.C., Kitano, H., Arkin, A.P., Bornstein, B.J., Bray, D., Cornish-Bowden, A., Cuellar, A.A., Dronov, S., Gilles, E.D., Ginkel, M., Gor, V., Goryanin, I., Hedley, W.J., Hodgman, T.C., Hofmeyr, J.H., Hunter, P.J., Juty, N.S., Kasberger, J.L., Kremling, A., Kummer, U., Noverre, N.L., Loew, L.M., Lucio, D., Mendes, P., Minch, E., Mjolsness, E.D., Nakayama, Y., Nelson, M.R., Nielsen, P.F., Sakurada, T., Schaff, J.C., Shapiro, B.E., Shimizu, T.S., Spence, H.D., Stelling, J., Takahashi, K., Tomita, M., Wagner, J., Wang, J.: The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* 19(4), 524–531 (2003)
13. Luciano, J.: PAX of mind for pathway researchers. *Drug Discovery Today* 10(13), 937–942 (2005)
14. Grafahrend-Belau, E., Weise, S., Koschützki, D., Scholz, U., Junker, B., Schreiber, F.: MetaCrop: a detailed database of crop plant metabolism. *Nucleic Acids Research* 36(Database issue), D954–D958 (2008)
15. Weise, S., Grosse, I., Klukas, C., Koschützki, D., Scholz, U., Schreiber, F., Junker, B.H.: Meta-All: a system for managing metabolic pathway information. *BMC Bioinformatics* 7(1), e465.1–9 (2006)

16. Varma, A., Palsson, B.: Metabolic Flux Balancing: Basic Concepts, Scientific and Practical Use. *Nature Biotechnology* 12(10), 994–998 (1994)
17. Kauffman, K., Prakash, P., Edwards, J.: Advances in flux balance analysis. *Current Opinion in Biotechnology* 14(5), 491–496 (2003)
18. Junker, B., Klukas, C., Schreiber, F.: VANTED: A system for advanced data analysis and visualization in the context of biological networks. *BMC Bioinformatics* 7(1), e109.1–13 (2006)
19. Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., Kummer, U.: COPASI—a COMplex PATHway SIMulator. *Bioinformatics* 22(24), 3067–3074 (2006)
20. Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M., Li, P., Oinn, T.: Taverna: a tool for building and running workflows of services. *Nucleic Acids Research* 34(Webserver issue), W729–W732 (2006)
21. McPhillips, T., Bowers, S., Zinn, D., Ludäscher, B.: Scientific workflow design for mere mortals. *Future Generation Computer Systems* 25(5), 541–551 (2009)

Towards Enhanced Retrieval of Biological Models through Annotation-Based Ranking

Dagmar Köhn¹, Carsten Maus², Ron Henkel¹, and Martin Kolbe¹

¹ University of Rostock

Institute of Computer Science, Database and Information Systems Group
Albert-Einstein-Str. 21, 18059 Rostock, Germany

{dagmar.koehn,ron.henkel,martin.kolbe}@uni-rostock.de

² University of Rostock

Institute of Computer Science, Modelling and Simulation Group
Albert-Einstein-Str. 21, 18059 Rostock, Germany

carsten.maus@uni-rostock.de

Abstract. Modelling and simulation methods gain increasing importance for the understanding of biological systems. The growing number of available computational models makes support in maintenance and retrieval of those models essential to the community. This article discusses which model information are helpful for efficient retrieval and how existing similarity measures and ranking techniques can be used to enhance the retrieval process, i. e. the model reuse. With the development of new tools and modelling formalisms, there also is an increasing demand for performing search independent of the models' encoding. Therefore, the presented approach is not restricted to certain model storage formats. Instead, the model meta-information is used for retrieval and ranking of the search result. Meta-information include general information about the model, its encoded species and reactions, but also information about the model behaviour and related simulation experiment descriptions.

Keywords: model storage, model retrieval, model reuse, annotation, ontologies, similarity, ranking.

1 Introduction

Modelling and simulation (M&S) is more and more becoming a standard technique for the study of complex biological systems. Systems biological models are formal descriptions of the interactions in a system, mostly describing quantitative dynamics so that the behaviour of the modelled system can be simulated over time. In particular for understanding the high non-linearity of biochemical systems, mathematical and computational models proved to be very helpful, e. g. to map out promising experiments in the wet laboratory by analysing simulation runs. With the growing number of developed models and their increasing complexity, model reuse becomes one of the major issues for modellers in the field of Systems Biology [14]. While the problems of model and information reuse are well-known and have already been discussed in great detail in the area of

M&S, many challenges still need to be faced [23], e.g. capturing the objective of a model. However, if one wants to start a modelling project to test a new hypothesis *in silico*, it would be desirable to take an existing model dealing with the same or a related system, which could be used as a basic starting point for modelling instead of re-writing everything from scratch. Also, to get a first idea of how a special biological phenomenon can be designed, e.g. an allosteric enzyme inhibition, considering previously developed models could enhance the model creation process in terms of time and quality.

Although many biological processes can be described by ordinary differential equations – and in fact most people do favor them for modelling [8] – other M&S techniques proved their necessity for Systems Biology as well, among them *process calculi*, *Petri Nets*, and *finite state automata*. To facilitate model exchange between users, databases and different simulation tools, machine-readable and standardised model representation formats have been developed. In the context of biological modelling, the most prominent and successful ones are the *Systems Biology Markup Language* (SBML) [7] and the *Cell Markup Language* (CellML) [4]. However, standards are always restrictive concerning their expressiveness and thus diverse formalisms are used to address different problems and needs of a modelling project. The sheer number of different modelling formalisms suggests that there will not be a single standard description language that covers every aspect of biological modelling, but specific standards with particular purposes will be used in the future [26].

As a consequence, one important step towards model reuse is a model retrieval system using an appropriate, formalism-independent storage format that still allows to elaborately search for relevant models. To achieve those goals, models are stored in a black-box manner, using available meta-information to gain knowledge about the modelled biological system. The feature of separating model and meta-information allows for efficient retrieval and comparison of models independent of the model encoding. Extracting additional information furthermore will enable a more sophisticated search. This paper proposes to use existing techniques from the research field of information integration to rank retrieved models regarding different similarity aspects. Although a solution for models of different biological levels is eligible, the main focus of this work is on the storage of biochemical cell models as those represent the majority of models developed today.

2 State of the Art

2.1 Model Annotations

With the growing demand for retrieval and reuse of biological models it became obvious that by only storing the pure model structure “most of the published quantitative models in biology are lost for the community because they are [...] insufficiently characterised to allow them to be reused” [16]. However, the need for model annotation and its realisation has also been a prominent discussion in the general field of M&S [27][20]. To motivate the use of annotations,

```

<listOfSpecies>
  <species metaid="_094854" id="aca" compartment="cell"
    initialConcentration="0.5" boundaryCondition="true"/>
  <species metaid="_094874" id="oaa" compartment="cell"
    initialConcentration="0.0003" boundaryCondition="true"/>
  [...]
</listOfSpecies>

```

Fig. 1. Example of an unannotated SBML model from the non-curated branch of the BioModels Database

Figure 1 shows an unannotated excerpt from a biological model description. The given XML snippet encodes a species. It makes clear how few information about the modelled system is provided by the XML code alone and, as a consequence, can be used for retrieval by just interpreting the XML elements. Who could tell what species or system the model is about?

One solution towards a more sophisticated description of models in terms of meta-information is the *MIRIAM* effort (Minimum Information Requested in the Annotation of Biochemical Models) [16]. It defines a standardised set of guidelines for the annotation, i.e. the supply of additional information, of biochemical models. *MIRIAM* guidelines can be applied to models and model elements of any structured formalism. Recommended information to be provided include the author of a model, the reference paper, and the meaning of the model itself as well as of the model constituents. The use of controlled vocabularies is suggested to encode those information. The according annotations are provided as triplets of form {“data-type,” “identifier,” “qualifier”} [16]. The identifier points to a certain piece of knowledge within a predefined data type; an example would be the identifier “GO:0000165” within the *MIRIAM* data-type “http://www.geneontology.org”. The qualifiers finally relate the referenced piece of knowledge with the according model element. Relations between a model or a model element and its annotation can vary from a simple *is* relation to *hasPart* or *isVersionOf*. Other qualifiers are deduced from typical biological terms, e.g. *isHomologTo* [13]. A good example for a model repository of annotated, *MIRIAM* compliant models, is the *BioModels Database* [15]. The XML example in Figure 2 shows the definition of a species that is annotated using the *Resource Description Framework* (RDF) [29] to specify that it concerns MAPK. Standardised (biological) qualifiers are used to define the relation between the species and its annotation, e.g. the sample annotation links to the resource <http://www.uniprot.org/#P26696> which points to the definition of the “Mitogen-activated protein kinase 1” in the *Unified Protein Resource Database* (Uniprot) [1] and uses the qualifier *is*. Another annotation links to the resource <http://www.uniprot.org/#P28482> using the qualifier *isHomologTo*, meaning that the species is not exactly the linked protein, but a homologous version of it, i.e. the human version of MAPK.

However, investigations also showed that storing meta-information according to the *MIRIAM* guidelines is not sufficient for an elaborated query on the


```

<species metaid="_584575" id="MAPK" name="MAPK" compartment="uVol"
  initialConcentration="280">
  <annotation>
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:bqbiol="http://biomodels.net/biology-qualifiers/"
      xmlns:bqmodel="http://biomodels.net/model-qualifiers/">
      <rdf:Description rdf:about="#_584575">
        <bqbiol:is>
          <rdf:Bag>
            <rdf:li rdf:resource="urn:miriam:uniprot:P26696"/>
          </rdf:Bag>
        </bqbiol:is>
        <bqbiol:isHomologTo>
          <rdf:Bag>
            <rdf:li rdf:resource="urn:miriam:uniprot:P28482"/>
          </rdf:Bag>
        </bqbiol:isHomologTo>
      </rdf:Description>
    </rdf:RDF>
  </annotation>
</species>

```

Fig. 2. Example of a species encoding with semantic annotations in the XML-based model description format SBML

meaning of models and [9] have suggested further “meaning facets” to be taken into account when describing the semantics of a model. Those include, for example, the behaviour of a model, the used formalism, or possible parametrisations during simulation. If envisioning the use of a model retrieval system for issues such as model comparison, composition, and integration, even much more detailed information has to be made explicitly available for queries [28], e.g. regarding model interfaces.

2.2 Model Repositories

For the storage of biological models, there already exist a number of model repositories [14]. Examples for publicly available SBML model repositories are the *BioModels Database* and the *JWS Online Model Database* [22]. Mathematical models encoded in CellML can be found in the *CellML repository* [18]. A database for models related to computational neuroscience and independent of a particular model encoding format is the *ModelDB* [19].

Looking at the different repositories, there are huge differences in how they abide by standardisation efforts and to what extent they support model retrieval. The *CellML repository* contains 386 models¹ encoded in CellML which can be searched by model type, e.g. cell cycle, signal transduction, or metabolism. Additionally, models of different curation states can be searched and also a full text search on

¹ last checked: 9th of April, 2009.

the models is offered. Search results are returned ordered by author. The *JWS Online Model Database* supports the search for SBML models by a limited number of keywords, including the author, publication title and journal, organism or model type. There is also a web-based tool offering a searchable categorisation of models in the JWS Online Model Database, distinguishing, for example, between cell cycle models and metabolism [22]. A full text search is not supported. Search results are returned ordered by author name. The *ModelDB* supports full text search as well as search for author name or accession number. The complete list of models can be retrieved sorted by the model name or by author. Additionally, some predefined queries regarding different criteria such as cell type or simulators are available. Models of any format can be stored. However, the most detailed search is offered by the *BioModels Database* which currently provides SBML representations of 211 curated and 124 non-curated models with together tens of thousands of reactions [2]. Full text search is supported and additionally search by species names, authors, biological publications and ontology entry terms is possible. Alternatively, one can browse through the Gene Ontology tree to find models for an ontology entry. All search results are returned sorted by model ID. To our knowledge, the BioModels Database offers most advanced retrieval techniques for biological models. It offers search for both, XML structure elements and annotations. Given a search term, the system returns not only models that contain the term in the XML code. Rather is the ontology term used to query the according ontology, taking into account parent and child nodes to perform the model search.

Some major drawbacks can be observed with existing databases for biological computational models: Currently, most of the model repositories in the biological application domain are restricted to a specific model description format. The limitation to a certain model encoding format could hamper model reuse, as in many cases the explicit formalism does not matter and, as a consequence, users have to check different databases, collect the search results and evaluate them manually to get a sufficiently complete overview of available models related to a specific topic. As already mentioned, the ModelDB for computational neuroscience models follows a formalism-independent storage approach though. However, model formats used in its application field typically just allow to annotate models using unstructured textual comments within the model code and automated meta-information extraction can not easily be applied. Furthermore, none of the previously described databases supports the ranking of search results. The retrieved models are either sorted alphabetically or by an arbitrary model ID, leading to time-consuming and tedious manual evaluations of the search results to find out which of the found models fit best to the modellers needs. Another problem of existing model repositories is the lack of a model version control. Currently, changes applied to published models stored within the model repositories cannot be tracked by different URLs or IDs. They might not even be stated on the web page. It is therefore impossible for users to notice changes, for example in the mathematics or parametrisation. This situation leads to serious problems, e. g. whenever working with a model through a reference.

² last checked: 9th of April, 2009.

3 A Database Schema for Annotated Black-Box Storage of Biological Models

To show the idea of determining model similarities across different model formalisms based on meta-information, we propose to store models unchanged in their according formats, i. e. in a black-box manner, as well as available additional information. Following a famous definition by Minsky, "a model (M) for a system (S) and an experiment (E) is anything to which E can be applied in order to answer questions about S." [21]. According to this definition, we do not only store the model description (M), but also meta-information about the modelled system (S) and related simulation experiment descriptions (E).

The EER diagram of the database schema shown in Figure 3 was designed to support various kinds of simple boolean queries, such as "Return all models of format type CellML." or "Return all models created by the author Goldbeter." It can also handle more complex queries, for example: "Return all multi-compartment models.", i. e. all models with more than one compartment, "Return all models encoding MAP Kinase cascade pathways in human.", or "Return all models where ATP is a reaction product.". Queries then might be combined to further restrict the result set: "Give me all models where ATP is a reaction product *and* where the model format is SBML *and* models not older than 2006."

Instead of returning an alphabetical list of models for a given query, the result set can be ranked regarding similarity characteristics. To do so, a simple similarity

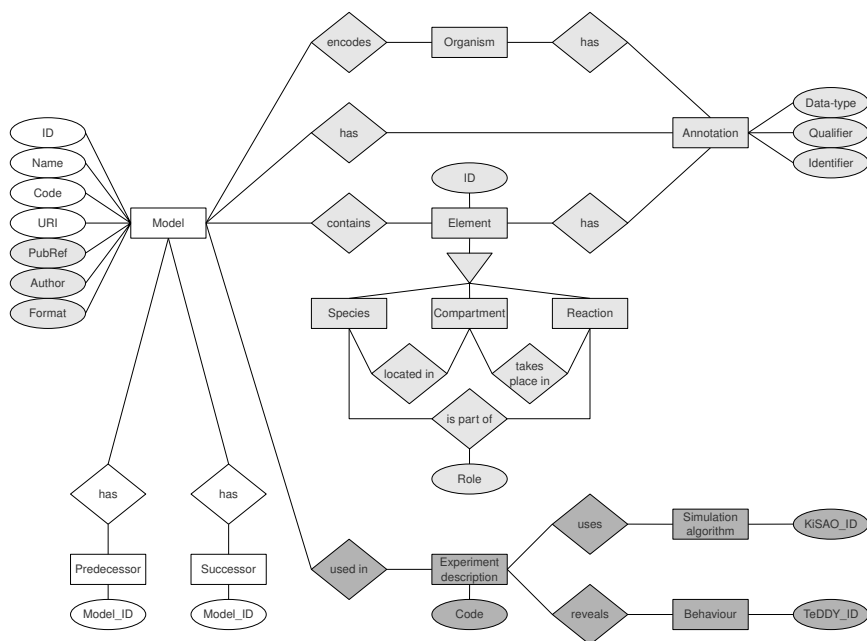


Fig. 3. Database schema: meta-information shown in light grey, information about experiments shown in dark grey. The default cardinality is m:n.

function with weighted features is suggested (see section 4). As the proposed similarity function builds on meta-information, those have to be extracted and stored separately, while the models themselves can be stored in a black-box manner. The advantage here is that the storage is independent of particular formats or formalism variants – we allow to store models in any format, even binary files. However, it is recommended to use existing XML exchange formats (e. g. SBML, CellML, PNML etc). Doing so enables the use of RDF annotations within the model descriptions. If then the models are uploaded into the model base, existing annotations can be extracted automatically, as has been introduced in [31], and the users' effort for providing necessary meta-information is reduced essentially. Additional and missing annotations, as well as annotations for non-XML-based models, might be added manually. Ideas on how to do that have been shown in [25].

To support the various queries and later result ranking, different types of information about the models have to be made available. The central entity of the design as shown in Figure 3 is the `Model` entity together with its `Name` and `ID` attributes. The `Code` attribute holds the model encoding and the (optional) `URI` points to the original model source. In addition, meta-information is kept on the model level and on the elements of a model. Meta-information on the model level include the ones specified in the MIRIAM guidelines, e. g. the model author (`Author`), or the publication reference describing the encoded model (`PubRef`). Further annotations, e. g. the model `Organism`, can be described in a generic `Annotation` entity. Those descriptions are typically realised as ontology references, often using more than one pair of qualifier and ontology reference. Examples for ontological annotations of a model and the according qualifier were already given in section 2. We propose capturing information to the version of a model by storing `Predecessor` and `Successor` model IDs as well. Model versions are of special importance for the model development process within larger research groups, but also for model integration and merging processes. For researchers referencing particular versions of one model, it has to be ensured that those versions will be accessible in the future. Furthermore, we also want to explicitly store the formalism the model is encoded in – here, at least the information about the format (`Format`), e. g. SBML, is required. But even more detailed information such as `SBML Level 2` can be stored. Therefore, it would be helpful to link to unique entries in an ontology of modelling formalisms, as was already proposed in [6].

Meta-information on the model element level include the description of species and reactions encoded in the model, as well as information on the model compartments. Those key elements of biochemical models can directly be extracted from some model description languages, e. g. SBML, if those are well-annotated. In addition to the specification of the three key entities of biochemical systems, the schema allows to describe their relations to each other, including which compartment a species occurs in or which reaction a species takes part in with what roles (reactant, product, modifier). These information can also be extracted from various model formats, e. g. SBML and Petri nets models.

What can be done if a formalism lacks an explicit definition of these elements? CellML models, e. g., are missing the reaction element due to directly encoding the differential equations for each species. As most of the Systems Biology models today focus on biochemical processes, the categorisation into compartments, species, and reactions still fits well and can be found on the meta-level, i. e. on the global knowledge of the model. Annotations for missing elements can be included for automated extraction, e. g. at the top of the model code, or they can be assigned manually. As covering all necessary meta-information about Systems Biology models a priori is an impossible task, the proposed schema allows to annotate other kinds of model elements in a general **ELEMENT** entity. Therefore, it is also possible to annotate further structures which are not part of typical biochemical models and also models of other biological levels can be properly described by suitable meta-information.

Last but not least, the necessary information on the context of the model includes information about possible experiment setups of the model. The final goal of retrieving a model in many cases is to be able to simulate it. That is why, it is important to keep information about valid simulation experiments. A format for the description of such simulation experiments is the *Simulation Experiment Description Markup Language* (SED-ML) [12]. Together with the SED-ML file, the model can be passed to a simulation tool and be simulated without any further parametrisation or adaptation. However, the storage of other experiment description formats is supported as well. Again, the favour for XML-based formats yields an additional and important advantage: changes on an XML-based reference model, e. g. in order to simulate a model with different parameter sets, can be defined by using XPath [30] expressions to address the according element that needs revision. This is a very native, precise and automatable way of describing model perturbations. As a consequence, retrieved models can be executed “on the fly” which is especially attractive for users who only want to study a system behaviour without understanding the model in full detail, as well as for users who might just need the simulation result as a starting point for their own approach. To facilitate the direct reuse of simulation experiment descriptions in the database, they are stored in the **Code** attribute of the **Experiment Description** entity. We also extract information about the simulation algorithm that can be applied to the simulation (**Simulation Algorithm**) from the experiment description. This will enable the reasoning on what types of simulations could be run on the model. For example, a model simulated with a Gillespie algorithm is capable of doing stochastic simulation experiments. To specify the simulation algorithm, we suggest to use the *Kinetic Simulation Algorithm Ontology* (KiSAO) [11]. KiSAO is an ontology of simulation algorithms commonly used in the field of but not restricted to Systems Biology. Additionally, information on the behaviour of the model under certain conditions, i. e. in a certain simulation study, is extracted and stored in the **Behaviour** entity. Here, we support the use of the *Terminology for the Description of Dynamics* (TeDDY) [10]. TeDDY is a controlled vocabulary for the description of the model behaviour for a certain set of parameter values. Future developments of simulation description

Table 1. Excerpt of meta-information annotations of a biochemical model

DB feature	Qualifier	Reference	Meaning
format	is	SBML_L2_V1	Systems Biology Markup Language (Level 2, Version 1)
author	is	vCard	Various author information
model	isDescribedBy isVersionOf isHomologTo	pubmed:10712587 go:0000165 reactome:react_634	Publication of the model MAP kinase kinase cascade MAP kinase cascade [Homo sapiens]
organism	is	taxonomy:8355	<i>Xenopus laevis</i>
compartment	is	go:0005623	Cell
species	is isHomologTo	uniprot:P26696 uniprot:P28482	Mitogen-activated protein kinase 1 Mitogen-activated protein kinase 1
species	is	uniprot:Q05116	Dual specificity mitogen-activated protein kinase kinase 1
reaction	isVersionOf isHomologTo isVersionOf	go:0000185 reactome:react_525 sbo:0000012	Activation of MAPKKK activity Stabilisation of RAF by further phosphorylation [Homo sapiens] Mass action rate law
reaction	isVersionOf isVersionOf isEncodedBy	go:0006468 sbo:0000216 sbo:0000012	Protein amino acid phosphorylation Phosphorylation Mass action rate law
behaviour	is	teddy:0000066	Periodic oscillation

formats hopefully enable to store more detailed information, for example, the designated tool and its detailed setup with which a specific model behaviour was observed.

Table 1 shows some examples for meta-information that is accessible from a well-annotated model. The first column holds the type of information that corresponds to the database design described above. The second column then contains the qualifiers which describe the type of relation between model elements or characteristics and their annotation, while the third column shows examples of references to meta-information and ontological resources (corresponding to the **Data-type** and **Identifier** attributes in the database schema in Figure 3). The fourth column gives a natural language description.

4 Determining Similarities between Biological Models

Once stored in the database, models can be retrieved from the repository by querying the extracted model information. Therefore, the annotations are fundamentally important to retrieve appropriate models. A query specified by the user consists of several values for the various search characteristics the sought model should contain. This so-called *feature matrix* results in the creation of a virtual query model which is compared with the stored models regarding its similarity. For our work, we use the three intuitions of [17] to define what similarity is:

Intuition 1: The similarity between A and B is related to their commonality. The more commonality they share, the more similar they are.

Intuition 2: The similarity between A and B is related to the differences between them. The more differences they have, the less similar they are.

Intuition 3: The maximum similarity between A and B is reached when A and B are identical, no matter how much commonality they share.

The function proposed for the estimation of similarity is shown in definition [□](#).

Definition 1. Let F be a non ordered set of features. Let M_X be a set representing a model containing feature values $\{x_{a_1}, \dots, x_{a_n}\}$ and M_Y be a set representing a model containing feature values $\{y_{b_1}, \dots, y_{b_n}\}$ with $\{a_1, \dots, a_n, b_1, \dots, b_n\} \in F$. Furthermore, $sim_{feature(i)}(x_i, y_i)$ is a function to compute the similarity for the feature $i \in F$.

$$similarity = \begin{cases} \frac{\sum^{\forall i \in F} (\alpha_i \times sim_i(x_i, y_i))}{\sum^{\forall i \in F} in(x_i, y_i)} & \text{if } M_X \cap M_Y \neq \emptyset \\ 0 & \text{else} \end{cases} \quad (1)$$

$$sim_i = \begin{cases} sim_{feature(i)}(x_i, y_i) & \text{if } x_i \in M_X \wedge y_i \in M_Y \\ 0 & \text{else} \end{cases} \quad (2)$$

$$in(x_i, y_i) = \begin{cases} 1 & \text{if } x_i \in M_X \wedge y_i \in M_Y \\ 0 & \text{else} \end{cases} \quad (3)$$

The similarity of the virtual query model M_X with features $(x_{a_1}, \dots, x_{a_n})$ and a model from the model repository M_Y with features $(y_{b_1}, \dots, y_{b_n})$ is determined by the sum of similarities of the features $sim_i(x_i, y_i)$ weighted with the importance of the feature α_i . For normalisation, the value is divided by the summation of features $in(x_i, y_i)$ which are the ones relevant in the query and present in the result model. For features based on ontological knowledge, the similarity ($sim_{feature(i)}(x_i, y_i)$) of a feature i in two different models M_X and M_Y is calculated on the result of the similarity estimation of the ontology terms, as well as on the evaluation of the according qualifiers. The remaining similarities are determined using string matching techniques, e. g. the Levenshtein distance.

The difficulty in ranking the search results lies within the fact that the understanding of *similarity* depends on the intention of the user: models can be similar when describing the same mathematics. But they could, on the contrary, also be found similar if encoding the same system – which does not necessarily mean that the models contain similar mathematics. To overcome this problem, we will provide the user with a number of pre-defined weighting functions. Furthermore, we also plan to enhance this functionality to allow for user-defined weighting functions.

While there exist various concepts for the estimation of similarities between strings [\[3\]](#), another challenge is the definition of the similarity function for ontological information. To determine the similarity of two features, the according ontology entries have to be matched. In the simplest case, both IDs come from

the same ontology and are identical. Then, the elements are considered equal. If two IDs are different and come from the same ontology, it has to be evaluated how both relate to each other. This is done using the path length between two ontology entries, as well as the entries' position (depth) in the ontology. A short path length can be expected if both entries are within the same concept. A high depth implies entries related to each other on a more specific layer. Siblings will be more similar than IDs from different ontology tree branches. However, the IDs might as well come from different ontologies. Here, matching techniques across ontologies, so called ontology alignments, have to be used to determine a similarity value. Related work on the topic can be found in [5].

4.1 Similarity Criteria for Biochemical Models

Different kinds of information about a model are already available by either evaluating its RDF annotations or by storing existing external, model related data. A first similarity estimation on the retrieved models can be done by examining the basic information provided, namely the *model name*, the *publication reference* and the *author of the model*. The meta-information about *species* and *reactions* taking place in the model are also very important as biological systems can be considered more alike if they encode similar biological problems. Additionally, the comparison of the encoded mathematics or transitions, i. e. the reactions, helps to identify similar models in the sense that they use similar strategies to depict the biological system. One example is the description of enzyme reaction kinetics in a model. It matters if they are described by Michaelis-Menten kinetics or detailed mass-action kinetics [2]. Another type of information that can be helpful to determine the similarity of certain models is the information on the simulation studies applied to the retrieved models. This includes information on the *simulation algorithm* that can be applied to the model to simulate it and the *behaviour* the model shows when simulated, e. g. an oscillation. Two models of the same biological system, returning similar types of result curves are more likely to encode the system with similar assumptions and mathematics than systems showing a different behaviour. And the outcome of the simulation of two models annotated with experiment descriptions using the same simulation algorithm can be expected more alike than if both models use different ones, especially if those algorithms have completely different characteristics, e. g. stochastic vs. deterministic.

4.2 Sample Search and Ranking

Assume that the user searches for models which (1) describe the metabolic pathway Glycolysis in (2) the baker's yeast (*Saccharomyces cerevisiae*), (3) are encoded in the Systems Biology Markup Language (SBML), and (4) show valid simulation results with the deterministic Livermore solver for ordinary differential equations (LSODE). These information can be mapped to four features in the feature matrix, namely "model"="Glycolysis", "organism"="S. cerevisiae", "format"="SBML", and "Simulation algorithm"="LSODE".

Table 2. Similarity example. The more similar a feature is between a model and the query, the darker is its background shading.

Query	model Glycolysis	format SBML	organism S. cerevisiae	simul. algorithm LSODE
Model m1	<i>hasPart</i> . Glycolysis	<i>is</i> .CellML	<i>is</i> .E. coli	<i>is</i> .LSODE
Model m2	<i>isVersionOf</i> . MAPK cascade	<i>is</i> .SPiM code	<i>is</i> .S. cerevisiae	<i>is</i> .Gillespie_SSA
Model m3	<i>is</i> .Glycolysis	<i>is</i> .SBML	<i>is</i> .S. cerevisiae	<i>is</i> .PVODE
Model m4	<i>is</i> .Glycolysis	<i>is</i> .SBML	<i>is</i> .T. brucei	–

Table 2 illustrate the idea of similarity measures using a small set of four models. Column “model” refers to the “Model_has_Annotation” relation of the database schema in Figure 3, “format” refers to the “Format” attribute of the Model entity, “organism” refers to the “Organism_has_Annotation” relation and “simul. algorithm” refers to the “Experiment_uses_Simulation Algorithm” relation of the database schema. The level of similarity between model feature and query is illustrated by grey shadings; the more similar a feature is between a model and the query, the darker is its background shading. As the implementation of a sophisticated similarity function is part of our future work, we use a simplified version of it for this example to show the basic idea:

$$sim_{feature(i)} = \begin{cases} 0 & \text{Features } (x_i, y_i) \text{ do not match} \\ 0.5 & \text{Features } (x_i, y_i) \text{ are related} \\ 1 & \text{Features } (x_i, y_i) \text{ match} \end{cases} \quad (4)$$

Compared to the table, dark grey stands for absolute match, light grey stands for partial match, and features with no similarity do not show any shading. For example, in the second column of table 2, the models m3 and m4 show dark grey marking because they are annotated with information that are totally similar to the query, i. e. they describe the Glycolysis pathway. Model m1 was also found to be matching the query. However, its annotation shows a weaker qualifier (“hasPart” instead of “is”) meaning that the model describes a larger system than the Glycolysis which is only a part of it. Therefore, the level of similarity is lower compared to models m3 and m4. Model m2 describes a completely different system (MAPK cascade). The next column shows the similarity for the feature “format”. The user requested a model encoded in the SBML format. Model m1 shows partial similarity, although the modelling formalism is not SBML but CellML. It is a legitimate assumption, because both SBML and CellML are formats to describe biological systems by mathematical equations. In contrast, the *Stochastic Pi-Machine* (SPiM) [24] model description language used in model m2 is quite different to SBML. To automate these “intuitive” decisions of partial similarity, we state that an ontology for the classification of modelling formalisms and formats would be desirable. For a more comprehensive discussion on how ontologies can facilitate modelling and simulation see [6]. The information for the organism feature of the models are

Table 3. Calculated similarity values

Query	model	format	organism	simul. algorithm	
<i>Feature weight</i>	Glycolysis	SBML	S. cerevisiae	LSODE	similarity
	<i>0.5</i>	<i>0.1</i>	<i>0.3</i>	<i>0.1</i>	
Model m1	0.25	0.05	0	0.1	0.025
Model m2	0	0	0.3	0	0.01875
Model m3	0.5	0.1	0.3	0.05	0.059375
Model m4	0.5	0.1	0.15	0	0.0625

shown in column four. While the models m2 and m3 both link to the queried organism, the other two models describe systems of different biological organisms. Here, model m4 is more similar to the query than m1, because both organisms *Trypanosoma brucei* and *Saccharomyces cerevisiae* belong to the eukaryota branch of the taxonomy tree, while *Escherichia coli* belongs to the bacterial kingdom of life and thus potentially shares less characteristics with the others. In the last column of table 2, an example of algorithms applicable to the model for simulation is given. The request asks for valid simulation experiments using a specific simulation algorithm for solving differential equations. Model m1 fulfils this need, model m3 partially does.

The overall evaluation of annotations in this example provides the ranked list of result models shown in table 3. The similarity values have been calculated using the similarity function described in equation 1. For the given example, we assumed the user has defined the following weights for the features considered for the query: model: 0.5, organism: 0.3, format: 0.1, simulation algorithm: 0.1. We also used the simplified similarity function as shown in equation 4. The final ranked result list can be shortened by setting a threshold for a minimum similarity value. One could also use the feature weights to determine “very important” features which must match at least partially. For example, model m2 describes a MAPK cascade and thus can be ignored for further similarity measures to the query of a Glycolysis model, even if the other features show high similarity. This holds because the feature weights indicate that the modelled system attribute is much more important than others, e.g. the format. At the end of the retrieval workflow, the user can assume that model m4 most likely fulfils his/her needs.

5 Conclusions

An integrated model storage and retrieval approach has been presented in this paper to show how similarity measures can be applied to models of biological systems. In contrast to most existing solutions, this approach allows to store models independent of the representation format and underlying modelling formalism. For the evaluation of database requests, it is of minor interest *how* a model encodes a biological component or reaction, but it is necessary to retrieve the information about *which* ones it encodes. The concept is based on the fact that models can be annotated with additional information which are mainly referring to biological ontologies and are encoded in the RDF format. Those

information can be used to define similarity measures for the result ranking of retrieved models. The work concentrates on the conceptual definition of similarities between biological models, making use of the introduced database to promote the application of similarity measures in existing model resources.

The considered features for similarity measures include meta-information on the level of the whole model, model elements, and simulation information. However, the identification of the meta-information qualified for satisfying ranking results is not fixed, but under steady discussion. To advocate the use of ranking, a simple similarity function based on a subset of the identified model features is shown. Later on, search results shall be ranked based on more sophisticated similarity functions. There exist a great number of methods and algorithms for ranking in the area of information retrieval and evaluations of various existing techniques are planned for the near future. Further investigations will show whether it is possible to apply existing algorithms stemming from multimedia document retrieval and retrieval of linked documents as to be found in the World Wide Web to the domain of Systems Biology. We also plan to realise user defined similarity weights in the future, which will result in user-specific ranking matrices. Another even more challenging example for user-specific ranking is the one of finding “compatible” models, where compatible is understood as “requires the same input” or as “the models can be coupled”. However, the question of model composition and coupling is a research task for its own and much work has to be done to support practical model reuse in this context [23,28]. Compatibility is therefore out of this works’ scope.

In summary, the introduced ideas can be considered as a first step towards enhanced model reuse in the field of Systems Biology. It proposes integrative storage, retrieval, and ranking of models through model annotations. A database design has been set up using the DB2 system and first experiments for similarity measures have been conducted. Future visions comprise a more sophisticated similarity measure that might ease model couplings. Elaborated testing on real life models are planned to test the database design for efficiency and performance.

Acknowledgements. This work is funded by the German research association (DFG) as part of the research training school dIEM oSiRiS. The authors thank Christian Knüpfer and Wolfram Liebermeister for fruitful discussions on model semantics and Lin Uhrmacher for helpful advice on general reuse problems in the field of modelling and simulation.

References

1. Bairoch, A., Apweiler, R., Wu, C.H., Barker, W.C., Boeckmann, B., Ferro, S., Gasteiger, E., Huang, H., Lopez, R., Magrane, M., Martin, M.J., Natale, D.A., O’Donovan, C., Redaschi, N., Yeh, L.-S.L.: The universal protein resource (uniprot). *Nucleic Acids Research* 33, 154–159 (2005)
2. Breitling, R., Gilbert, D., Heiner, M., Orton, R.: A structured approach for the engineering of biochemical network models, illustrated for signalling pathways. *Briefings in Bioinformatics* 9(5), 404–421 (2008)

3. Cormen, T.: Introduction to algorithms. MIT Press, Cambridge (2001)
4. Cuellar, A.A., Lloyd, C.M., Nielsen, P.F., Bullivant, D.P., Nickerson, D.P., Hunter, P.J.: An overview of cellml 1.1, a biological model description language. *SIMULATION* 79(12), 740–747 (2003)
5. Ehrig, M., Haase, P., Hefke, M., Stojanovic, N.: Similarity for ontologies - a comprehensive framework. In: Workshop Enterprise Modelling and Ontology: Ingredients for Interoperability, at PAKM 2004 (2005)
6. Fishwick, P., Miller, J.: Ontologies for modeling and simulation: issues and approaches. In: Proceedings of the 2004 Winter Simulation Conference, pp. 259–264 (2004)
7. Hucka, M., Finney, A., Sauro, H., Bolouri, H., Doyle, J.C., Kitano, H., et al.: The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* 19(4), 524–531 (2003)
8. Klipp, E., Liebermeister, W., Helbig, A., Kowald, A., Schaber, J.: Systems biology standards—the community speaks. *Nature Biotechnology* 25(4), 390–391 (2007)
9. Knüpfer, C., Beckstein, C., Dittrich, P.: Towards a semantic description of bio-models: Meaning facets - a case study. In: Proceedings of the Second International Symposium on Semantic Mining in Biomedicine, June 2006, pp. 97–100 (2006)
10. Knüpfer, C., Le Novère, N.: Teddy - a terminology for the description of the dynamics of bio-models. Poster, Foundations of Systems Biology and Engineering (2007)
11. Köhn, D., Le Novère, N.: The kinetic simulation algorithm ontology (KiSAO). Web Site (2007), <http://www.ebi.ac.uk/compneur-srv/kisao/>
12. Köhn, D., Le Novère, N.: SED-ML – an XML format for the implementation of the MIASE guidelines. In: Heiner, M., Uhrmacher, A.M. (eds.) CMSB 2008. LNCS (LNBI), vol. 5307, pp. 176–190. Springer, Heidelberg (2008)
13. Laibe, C., Le Novère, N.: MIRIAM Resources: tools to generate and resolve robust cross-references in Systems Biology. *BMC Systems Biology* 1(1), 58 (2007)
14. Le Novère, N.: Model storage, exchange and integration. *BMC Neuroscience* 7(suppl. 1) (2006)
15. Le Novère, N., Bornstein, B., Broicher, A., Courtot, M., Donizelli, M., Dharuri, H., Li, L., Sauro, H., Schilstra, M., Shapiro, B., Snoep, J.L., Hucka, M.: Biomodels database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Res.* 34(Database issue) (January 2006)
16. Le Novère, N., Finney, A., Hucka, M., Bhalla, U.S., Campagne, F., Collado-Vides, J., Crampin, E.J., Halstead, M., Klipp, E., Mendes, P., Nielsen, P., Sauro, H., Shapiro, B., Snoep, J.L., Spence, H.D., Wanner, B.L.: Minimum information requested in the annotation of biochemical models (MIRIAM). *Nature Biotechnology* 23(12), 1509–1515 (2005)
17. Lin, D.: An information-theoretic definition of similarity. In: ICML 1998: Proceedings of the Fifteenth International Conference on Machine Learning, San Francisco, CA, USA, pp. 296–304. Morgan Kaufmann Publishers Inc., San Francisco (1998)
18. Lloyd, C.M.M., Lawson, J.R.R., Hunter, P.J.J., Nielsen, P.F.F.: The cellml model repository. *Bioinformatics* (July 2008)
19. Migliore, M., Morse, T.M., Davison, A.P., Marenco, L., Shepherd, G.M., Hines, M.L.: Modeldb: making models publicly accessible to support computational neuroscience. *Neuroinformatics* 1(1), 135–139 (2003)
20. Miller, J., Baramidze, G.: Simulation and the semantic web. In: Proceedings of the 37th Winter Simulation Conference, pp. 2371–2377 (2005)

21. Minsky, M.: Models, minds, machines. In: Proceedings of the IFIP Congress, pp. 45–49 (1965)
22. Olivier, B.G., Snoep, J.L.: Web-based kinetic modelling using jws online. *Bioinformatics* 20(13), 2143–2144 (2004)
23. Overstreet, C., Nance, R., Balci, O.: Issues in enhancing model reuse. In: International Conference on Grand Challenges for Modeling and Simulation, pp. 27–31 (2002)
24. Phillips, A., Cardelli, L.: Efficient, correct simulation of biological processes in the stochastic pi-calculus. In: Calder, M., Gilmore, S. (eds.) CMSB 2007. LNCS, vol. 4695, pp. 184–199. Springer, Heidelberg (2007)
25. Schulz, M., Uhlenhof, J., Klipp, E., Liebermeister, W.: SBMLmerge, a system for combining biochemical network models. In: Genome informatics. International Conference on Genome Informatics, vol. 17(1), pp. 62–71 (2006)
26. Strömbäck, L., Hall, D., Lambrix, P.: A review of standards for data exchange within systems biology. *Proteomics* 7(6), 857–867 (2007)
27. Tolk, A., Muguira, J.: The Levels of Conceptual Interoperability Model. In: Proceedings of the 2003 Fall Simulation Interoperability Workshop (2003)
28. Uhrmacher, A., Degenring, D., Lemcke, J., Kraemer, M.: Towards reusing model components in systems biology. In: Danos, V., Schachter, V. (eds.) CMSB 2004. LNCS (LNBI), vol. 3082, pp. 192–206. Springer, Heidelberg (2005)
29. W3C. Resource Description Framework (1997), <http://www.w3.org/RDF>
30. W3C. XPath (November 1999), <http://www.w3.org/TR/xpath>
31. Wang, T., Wang, J., Yu, Y., Shen, R., Liu, J., Chen, H.: Metadata pro: Ontology-based metadata processing for web resources. In: Bussler, C.J., Hong, S.-k., Jun, W., Kaschek, R., Kinshuk, Krishnaswamy, S., Loke, S.W., Oberle, D., Richards, D., Sharma, A., Sure, Y., Thalheim, B. (eds.) WISE 2004 Workshops. LNCS, vol. 3307, pp. 34–45. Springer, Heidelberg (2004)

Author Index

- Agrawal, Gagan 141
Ang, Wee Tiong 127
Baker, Christopher J.O. 127
Cadag, Eithon 55
Campi, Alessandro 88
Ceri, Stefano 88
Chelliah, Vijayalakshmi 5
Colmsee, Christian 196
Dada, Joseph O. 182
Endler, Lukas 5
Froidevaux, Christine 113
Gibrat, Jean-François 113
Gibson, Toby 173
Grafahrend-Belau, Eva 196
Gross, Anika 71
Hallinan, Jennifer 31
Hartung, Michael 71
Haslam, Niall 173
Henkel, Ron 204
Hindle, Matthew 16
Hossain, Shahriyar 157
Ives, Zachary G. 1
James, Katherine 31
Jamil, Hasan 157
Junker, Björn 196
Juty, Nick 5
Kanagasabai, Rajaraman 127
Kirsten, Toralf 71
Klukas, Christian 196
Klyne, Graham 47
Köhler, Jacob 16
Köhn, Dagmar 204
Kolbe, Martin 204
Laibe, Camille 5
Lambrix, Patrick 127
Lange, Matthias 196
Laurila Bergman, Jonas 127
Le Novère, Nicolas 5
Li, Chen 5
Liu, Tantan 141
Lysenko, Artem 16
Masseroli, Marco 88
Maus, Carsten 204
Mendes, Pedro 182
Miles, Alistair 47
Mir, Saqib 96
Myler, Peter J. 55
Rahm, Erhard 71
Rance, Bastien 113
Rawlings, Christopher J. 16
Rodriguez, Nicolas 5
Rojas, Isabel 96
Scholz, Uwe 196
Schreiber, Falk 196
Shotton, David 47
Staab, Steffen 96
Tarczy-Hornoch, Peter 55
Taubert, Jan 16
Wang, Fan 141
Weile, Jochen 16
Weise, Stephan 196
Wipat, Anil 31
Zhao, Jun 47