

Christopher J.O. Baker  
Greg Butler  
Igor Jurisica (Eds.)

LNBI 7970

# Data Integration in the Life Sciences

9th International Conference, DILS 2013  
Montreal, QC, Canada, July 2013  
Proceedings

 Springer

# Lecture Notes in Bioinformatics

7970

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

Christopher J.O. Baker Greg Butler  
Igor Jurisica (Eds.)

# Data Integration in the Life Sciences

9th International Conference, DILS 2013  
Montreal, QC, Canada, July 11-12, 2013  
Proceedings



Springer

## Volume Editors

Christopher J.O. Baker  
University of New Brunswick  
Department of Computer Science and Applied Statistics  
Saint John, NB E2L 4L5, Canada  
E-mail: bakerc@unb.ca

Greg Butler  
Concordia University  
Department of Computer Science and Software Engineering  
Montreal, QC H3G 1M8, Canada  
E-mail: gregb@encs.concordia.ca

Igor Jurisica  
University of Toronto  
Ontario Cancer Institute  
Toronto, ON M5G 1L7, Canada  
E-mail: juris@ai.utoronto.ca

ISSN 0302-9743 e-ISSN 1611-3349  
ISBN 978-3-642-39436-2 e-ISBN 978-3-642-39437-9  
DOI 10.1007/978-3-642-39437-9  
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013941943

CR Subject Classification (1998): H.3, J.3, I.2, H.4, C.2, H.2, H.5

LNCS Sublibrary: SL 8 – Bioinformatics

© Springer-Verlag Berlin Heidelberg 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

*Typesetting:* Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)



# Preface

DILS was established in 2004 as a new bioinformatics workshop focusing on topics related to data management and integration. Now in its ninth year the conference continues to attract researchers from across a range of disciplines all of whom recognize the challenges faced by life scientists in managing and reusing data. Stakeholders involved in digital ecosystems and data ownership are able to generate large volumes of high-quality data and want to publish it to the widest possible audience for prospecting by scientists. And yet, data are not knowledge. The real value is the translation of the data into actionable knowledge. The methodologies and frameworks we depend on to facilitate this translation are still evolving, and the challenges in data management and reuse have grown rather than diminished over the last decade and are common across many disciplines.

Life science remains one of the leading domains and continues to create massive amounts of diverse data needing validation, curation, and annotation with meaningful descriptions and formatting according to open standards to ensure it is sharable between interoperable distributed systems and accessible by end users. Practitioners are, however, continually experimenting and the forum for discussing which methodologies have succeeded, which new technologies are now being adopted, for which particular tasks, and how they are used to integrate data for subsequent bioinformatic analysis is DILS.

This year, DILS received 23 papers to the main research track (both long and short papers). Four papers were accepted unconditionally. A further six were accepted with the provision that authors revised their papers in accordance with reviewers' comments and provided detailed and itemized responses. All papers were subsequently verified by the Program Committee (PC) Chair and General Chairs.

Accepted papers cover a range of important topics including: algorithms for ontology matching, interoperable frameworks for text mining using Semantic Web services, pipelines for genome-wide functional annotation, automation of pipelines providing data discovery and access to distributed resources, knowledge-driven querying-answer systems, prisms, nanopublications, electronic health records and linked data. This year we opted to also offer an Early Career and Systems Track at the DILS workshop. At the time of writing, papers submitted to each track were still under review. These papers are not published in the research track proceedings.

DILS 2013 featured two keynote speakers. Firstly, Dr. Erich Gombocz, co-founder and CSO of IO Informatics a decade ago, is a veteran in applying systems biology approaches to pharmaceutical and clinical decision making based on semantic data integration and knowledge management technologies. Dr. Gombocz

presented the rationale for rethinking old problems, retooling with new methodologies and revisiting the process models that underpin our existing knowledge discovery in pharma and clinical practice in healthcare. Specifically, he advocates new patient-centric, precision-medicine healthcare models to change how drugs are developed, how trials are performed, and how patients are treated. His manuscript is included in the proceedings. Our second keynote speaker was Dr. Paolo Ciccarese, Assistant in Neuroscience at Massachusetts General Hospital and Instructor in Neurology at Harvard Medical School. He is known for his pioneering work on the Annotation Ontology, an RDF model for exchanging annotation. In his talk, Dr. Ciccarese introduced annotation as a form of “micro-integration,” in which typed, versioned, and provenance links are assigned between text and schema, text and data, or data and data. He showed how the Open Annotation standard facilitates both short- and longer-term data integration efforts, transforming content into smart and connected data.

DILS 2013 was held at Concordia University in Montreal, Canada, and was organized as part of a series of three co-located events known as the Semantic Trilogy. The two co-located events were the 4th International Conference on Biomedical Ontology and the 4th Canadian Semantic Web Symposium.

As the event co-chairs and editors of this volume, we would like to thank all authors who submitted papers, as well as the PC members and additional referees for their excellent work in evaluating the submissions. Special thanks go to Concordia University for providing us with the facilities to run the event, and the Semantic Trilogy organization team. Finally, we would like to thank Alfred Hofmann and his team at Springer for their cooperation and help in putting this volume together.

May 2013

Christopher J.O. Baker  
Greg Butler  
Igor Jurisica

# Organization

## General Chairs

Christopher J.O. Baker      University of New Brunswick, Saint John,  
Canada  
Greg Butler      Concordia University, Montreal, Canada

## Program Committee Chair

Igor Jurisica      University of Toronto, Canada

## Program Committee

Adam Lee      University of Maryland, USA  
Adrien Coulet      Loria - INRIA Nancy Grand Est, France  
Amar K. Das      Stanford University, USA  
Artjom Klein      University of New Brunswick, Canada  
Asif M. Khan      National University of Singapore, Singapore  
Bastien Rance      NIH, Bethesda, USA  
Brad Malin      Vanderbilt University, USA  
Christian Schönbach      Kyushu Institute of Technology, Japan  
David De Roure      Oxford e-Research Center, UK  
Dietrich Rebholz-Schuhmann      EBI, UK  
Erhard Rahm      University of Leipzig, Germany  
Fatima Al-Shahrour      Broad Institute of MIT and Harvard, USA  
Fleur Mougín      University of Bordeaux Segalen, France  
Guo-Qiang Zhang      Case Western Reserve University, USA  
Hasan Jamil      University of Idaho, USA  
James Cimino      NIH/CC/OD, USA  
Jörg Hakenberg      Arizona State University, USA  
Jerven Bolleman      Swiss Institute of Bioinformatics, Switzerland  
Jong Park      Korea Advanced Institute of Science and  
Technology, Korea  
Karen Eilbeck      University of Utah, USA  
Karin Verspoor      National ICT, Australia, NISTA  
Lawrence Hunter      University of Colorado, USA  
Marco Masseroli      Politecnico di Milano, Italy  
Marco Roos      LUMC, University of Amsterdam,  
The Netherlands  
Maria Esther Vidal      Universidad Simón Bolívar, Venezuela  
Matthew Hindle      Synthetic and System Biology, Edinburgh, UK

## VIII Organization

Michael Krauthammer	Yale University, USA
Mong Li Lee	National University of Singapore, Singapore
Neil Sarkar	University of Vermont, USA
Nigam Shah	Stanford Center for Biomedical Informatics Research, USA
Paolo Missier	Newcastle University, UK
Paolo Romano	Institute of Genoa, Italy
Peter Mork	Noblis, USA
Radhakrishnan Nagarajan	University of Kentucky, USA
Rainer Winnenbarg	NIH, USA
Satya Sahoo	Case Western Reserve University, USA
Tammy Cheng	Cancer Research UK London, UK
Vasant Honavar	Iowa State University, USA

## DILS Steering Committee

Sarah Cohen-Boulakia	LRI, University of Paris-Sud 11, France
Graham Kemp	Chalmers University of Technology, Sweden
Ulf Leser	Humboldt-Universität zu Berlin, Germany
Paolo Missier	Newcastle University, UK
Norman Paton	University of Manchester, UK
Louïqa Raschid	University of Maryland, USA
Erhard Rahm	University of Leipzig, Germany

## Organizing Committee

Greg Butler	Concordia University, Montreal, Canada
Christopher J.O. Baker	University of New Brunswick, Saint John, Canada
Michel Dumontier	Carleton University, Ottawa, Canada

## Webmaster

Artjom Klein	University of New Brunswick, Saint John, Canada
--------------	--

# Table of Contents

Changing the Model in Pharma and Healthcare – Can We Afford to Wait Any Longer? .....	1
<i>Erich Alfred Gombocz</i>	
Ibidas: Querying Flexible Data Structures to Explore Heterogeneous Bioinformatics Data .....	23
<i>Marc Hulsman, Jan J. Bot, Arjen P. de Vries, and Marcel J.T. Reinders</i>	
From Questions to Effective Answers: On the Utility of Knowledge-Driven Querying Systems for Life Sciences Data .....	38
<i>Amir H. Asiaee, Prashant Doshi, Todd Minning, Satya Sahoo, Priti Parikh, Amit Sheth, and Rick L. Tarleton</i>	
OmixAnalyzer – A Web-Based System for Management and Analysis of High-Throughput Omics Data Sets .....	46
<i>Thomas Stoltmann, Karin Zimmermann, André Koschmieder, and Ulf Leser</i>	
The RDF Pipeline Framework: Automating Distributed, Dependency-Driven Data Pipelines .....	54
<i>David Booth</i>	
Towards Interoperable BioNLP Semantic Web Services Using the SADI Framework .....	69
<i>Ahmad C. Bukhari, Artjom Klein, and Christopher J.O. Baker</i>	
Optimizing Similarity Computations for Ontology Matching - Experiences from GOMMA .....	81
<i>Michael Hartung, Lars Kolb, Anika Groß, and Erhard Rahm</i>	
Semi-automatic Adaptation of Mappings between Life Science Ontologies .....	90
<i>Anika Groß, Julio Cesar Dos Reis, Michael Hartung, Cédric Pruski, and Erhard Rahm</i>	
Next Generation Cancer Data Discovery, Access, and Integration Using Prizms and Nanopublications .....	105
<i>Jamie P. McCusker, Timothy Lebo, Michael Krauthammer, and Deborah L. McGuinness</i>	

Putting It All Together: The Design of a Pipeline for Genome-Wide Functional Annotation of Fungi in the Modern Era of “-Omics” Data and Systems Biology . . . . .	113
<i>Greg Butler</i>	
Mining Anti-coagulant Drug-Drug Interactions from Electronic Health Records Using Linked Data . . . . .	128
<i>Jyotishman Pathak, Richard C. Kiefer, and Christopher G. Chute</i>	
<b>Author Index</b> . . . . .	141

# Changing the Model in Pharma and Healthcare – Can We Afford to Wait Any Longer?

Erich Alfred Gombocz\*

IO Informatics Inc., Berkeley, California, USA  
egombocz@io-informatics.com

**Abstract.** Innovations in healthcare delivery and Pharma require re-examination of process models at the foundation of our knowledge discovery and clinical practice. Despite real-time availability of ‘big data’ from ubiquitous sensors, mobile devices, 3D printing of drugs, and a mind shift in data ownership, data integration still remains one of the core challenges to innovation. Increasingly persistent, semantic data integration is gaining recognition for its dynamic data model and formalisms which make it possible to infer from and reason over interconnected contextualized data, creating actionable knowledge faster and at lower cost. While such technical advances underpin the successful strategies to drive positive patient outcomes or accelerate drug design, there are equally profound social changes towards the willingness of patients to share their own data - opening doors to new patient-centric, precision-medicine healthcare models. Adding astronomically rising costs in research and healthcare, we have arrived at a critical turning point where it is now well within our reach to change how drugs are developed, how trials are performed and how patients are treated - and we can do this with huge benefits for otherwise unsustainable industries. Examples show that not only is this possible today, but that such approaches already have traction; (i) in Pharma for assessing impact of excipient on drug stability and efficacy; for pre-clinical toxicity assessment and integral systems views on drug safety, (ii) in Government at the FDA’s cross species biomarker initiative to reduce animal testing and (iii) in Health Care for organ transplant rejection assessment and COPD. Using comparative effectiveness and side effect analyses to base treatments on solid prognoses and therapy decision support, we can and must change discovery and healthcare into a data driven and patient centric paradigm. The socio-economic benefits of such a change will be enormous.

**Keywords:** life sciences, big data, sensors, data ownership, semantic integration, actionable knowledge, patient centric, precision medicine, decision support, use cases, socio-economics.

---

\* Corresponding author.

# 1 Introduction

## 1.1 Historic Models in Life Sciences

In the past, widespread utilization of large shared spreadsheets, dedicated laboratory information management systems (LIMS), large relational data warehouses and traditional methods for extraction, translation and loading (ETL) have been used across the life sciences enterprise spectrum with more or less sophisticated approaches to interconnect in-between some of those resources [1]. Key features of LIMS include acquisition, workflow and data tracking across different modalities, data exchange interfaces, audit functions and support for their use in regulated environments. Because of rapid pace at which laboratories and their data management needs shift, the definition of LIMS has become more blurred. This is particularly due to the fact that the needs of laboratories widely vary which requires also a shift in functionality of laboratory information management systems.

Historically, LIMS and process execution have performed similar functions, building an organization's reference backbone for experimental results. More recently, assay and ELN functions have been added to extend traditional LIMS systems. However, the need to implement quality standards, the awareness of data management solutions using different architectures and the unavailability of adapted solutions for interoperability led in many cases to in-house developments instead of using commercial solutions. Particularly in large Pharma organizations the separation of data into target areas, specific projects as well as the separation of R&D chemistry, assay development and biology caused limited communication in-between groups, redundant efforts and no integral view across the data. The strict separation between pre-clinical, clinical and market data has hampered feedback within the organizations to learn from past experiences. Consequently, adverse effects got missed; clinical trial efficiency was at a low point and causing a hesitant approach in the development of new drugs.

Despite ever rising amounts of data through high throughput screening, multiplexed assays and broad use of chip technologies, the actual knowledge produced in comparison to research costs was declining rapidly [2]. Large Pharma companies were buying their libraries of new compounds from small biotech to cut costs by reducing in-house research to small focus areas. Collaboration models were restricted to consortia with narrow goals and small portions of pre-clinical, pre-competitive segments the sharing party deemed to be of no further usefulness to the organization.

## 1.2 Rise of New Technologies and Machines

The data landscape changed with the rise of new technologies, new developments in instrumentation, automation and exponential increase in throughput of previously labor-intensive and time-consuming procedures. In the last several years, massive next generation sequencing (NGS), progress in whole genome sequencing using de novo assemblies on unimaginable scale [3], RNA sequencing and genome-wide association



studies (GWAS) have been at the forefront of genomics to be used for both, gene-based biomarker discovery and personal genomics as tool for precision medicine and the genetic selection of population cohorts for clinical trials.

The development of new sensor technologies and advances in mobile computing led to sensors being everywhere and on everything with real-time internet connectivity. Wearable medical technology is becoming a hot commodity [4]. As these devices come to market, they have great potential to help both patients and clinicians monitoring vital signs and symptoms [5]. In personal health, sensors which are always on, always with you, always tracking were changing data collection to become a continuous monitoring stream [6], providing both, individuals and physicians more accurate and more detailed data about many influence parameters on a health or disease state which previously were not available [7]. Lifestyle choices, such as exercises, habits and environments have been recorded similarly [8].

The size of all these data and the computational considerations to analyze them along with the high data dynamics require investment in High Performance Computing (HPC) and have led to tradeoffs between inexpensive highly dense storage on commodity disks and higher cost better performant NAS, SAN or Cloud services (CEPH, OpenStack, Amazon). Dependent on budgets, compromises were made, and raw data have been thrown out in favor of much smaller analyzed data sets. Algorithmic transformations to normalize in-between platforms have changed over time and metadata not always included, making review for verification in many cases impossible. While new ways of computing have been introduced which are using massive parallel computing and distributed clusters for analysis [9], management of 'big data' has become a complex, expensive and demanding task at scales beyond most forecast expectations. This development has created a new bottleneck in analysis and practical use of ever growing data repositories and made interoperability, provenance and versioning an equally important concern to plain connectivity and was instrumental in rethinking data integration in life sciences in general.

### **1.3 Economic Importance of Data**

In a MIT Technology Review end of 2012, the question was raised publicly if personal data is the new currency [10]. The economic importance of access and utilization of vast amounts of interconnected data can no longer be denied, and the same applied equally to the life sciences. With the expansion of social networks and the drive from individuals to take care of their needs for better prognosis and treatment, the frustration about public availability of medical data has driven the movement of patients making their own data publicly accessible. Adding to this the fact, that big data analytics became a way of turning data into money [11], the assessment that Data is the new money and those who have access to it, have power became obvious with significant implications in the shift from revenue and margins driven industrial models towards customer-centric, health outcomes for patients motivated strategies in which consumers influence the drivers of healthcare systems.

This new proposition is affecting insurers, payers, service providers, drug discovery processes, drug development, repurposing of drugs for new indications alike – moving to an evidence-based, outcomes-focused, behavior-driven life sciences environment which through empowering of data will benefit all of us.

## 1.4 Staying Competitive

In the past, low ROI on research and development has provided little incentives to innovation or change, particularly as industry was closely watching its competitor's moves to decide about the necessity to adjust their model to new trends based on proof-of-concept (PoC) and pilot study outcomes. In a way, the unwillingness to share even pre-competitive or failed approaches within tightly controlled consortia members for collaborations has reduced the competitiveness of the industry – however, there are several positive examples on the horizon that this behavior is changing as both, Pharma and Healthcare industry have come to the realization that everyone profits from collaborative approaches to accumulative and complimentary data on common goals. Of course, any meaningful collaboration is closely tied to interoperability, and this is true equally for both, commercial and academic entities.

Crowd-sourced analysis requires interoperability, and interoperability is how big data becomes big open data, and this will assure rapid progression in scientific discovery and providing a solid foundation for Pharma and healthcare to stay competitive. Acknowledged, that crowd-sourcing as a new policy has many implications [12-13] and there is still hesitancy to collaborative data sharing, its driving force will be costs and efficiency towards new concepts which will significantly shape the future of data-driven life sciences.

## 2 State of the Industry

### 2.1 Data Generation vs. Knowledge Gain

While automation and advances in technologies have brought down costs of complex testing faster than anticipated, analysis and integration towards applicable knowledge has lagged behind. Massive data (a single run of Life Technologies sequencer produces ~900 GB raw data, 1 machine = 10 TB/day) and the sophistication required by the complexity of analysis procedures have led to the notion of the “*\$1,000.-genome at \$1 Mio interpretation.*”[14] – thus, in most cases, only a small fraction of information is used to build knowledge and advance scientific progress.

One major reason for this discrepancy is that the required proficiency of a whole range of experts including molecular and computational biologists, geneticists, pathologists and physicians with detailed knowledge of disease and treatment modalities, genetic counsellors and IT specialists to build analysis teams [15] is not easy to establish. Using large number of specialists was critical to complete the data analysis, for variant annotations and to interpret causative or actionable variants. Even then, clinical verification of such variants and ramifications for the treating physician and patient require even today immense efforts and make the widespread use of

clinical whole-genome sequencing for diagnosis and quality of life improvement still a distant goal. On top of this, the bioethics pros and cons of WGS of every newborn child need to be sorted out and genomics must address socio-economic disparities in healthcare.

Similar considerations should be applied to other rapidly evolving fields such as proteomics, transcriptomics and microbial implications in major diseases – rapid data generation through automated, low cost high throughput sample analyses does not match up with the possible knowledge gain from those resources. Amongst other reasons, standardization of analytical methods and algorithms and requirements for quality standards on data also has played a significant role in the usability of results across laboratories.

## **2.2 Traditional Data Mining**

Relational data warehouses and object data bases require upfront considerations to determine which questions you want to answer, Data models (schemas) must be defined at the beginning, so such solutions, while excellent for final datasets and great performing on optimized queries for what they were built for are demanding in support due to their rigid and static structure. On the other hand, a whole host of mining solutions and visualization tools are available as relational database technology has been around for a long time and big players in information technology have embraced its use.

In Life Sciences, however, a different picture emerges as dynamic, agile solutions are required to keep pace with changing data types, formats, instrumentation as well as analytical requirements. Add to this the scale of growth, and ‘big data’ has demonstrated impressively, that more relational data warehouses and traditional data mining approaches cannot be the answer to today’s information requirements landscape. In many cases in Life Sciences, questions to ask and potential use cases are moving targets, so any inflexible solution limits its applicability. In biological systems, the need to traverse data, to infer from other data and to search complex pattern across all your resources to find clues what kind of questions you can answer is rooted on a different set of requirements - in most cases, questions are not predefined, and the picture what and how to ask is not clear at the beginning. Include to this that in the relational world no clear connections in-between data silos exist and different proprietary schemas prevent cross-resource queries, and the limitations of such approaches become transparent.

## **2.3 Cost of Research vs. Outcomes**

A recently published Forbes report [16] on staggering costs of new drugs, a new study comparing healthcare cost in the US with other countries [17] and the OECD Health Statistics [19] provide insights into costs of research versus outcomes which are stunning, but well known within the industry. A representative of Eli Lilly estimated the average cost of bringing a new drug to market at \$1.3 billion, a price that would buy 371 Super Bowl ads on television [16]. On average, a drug developed by a major pharmaceutical company costs at least \$4 billion in R&D (see Table 1 below)

**Table 1.** Number of approved drugs and drug development costs of major Pharma companies (2012)

Company	Approved Drugs	R&D Costs/Drug [\$ Mio]
AstraZeneca	5	11,790.93
GlaxoSmithKline	10	8,170.81
Sanofi	8	7,909.26
Roche AG	11	7,803.77
Pfizer Inc.	14	7,727.03
Johnson & Johnson	15	5,885.65
Eli Lilly & Co.	11	4,577.04
Abbott Laboratories	8	4,496.21
Merck & Co Inc	16	4,209.99
Bristol-Myers Squibb Co.	11	4,152.26
Novartis AG	21	3,983.13
Amgen Inc.	9	3,692.14

Source: InnoThink Center For Research In Biomedical Innovation; Thomson Reuters Fundamentals via FactSet Research Systems (adapted from [16])

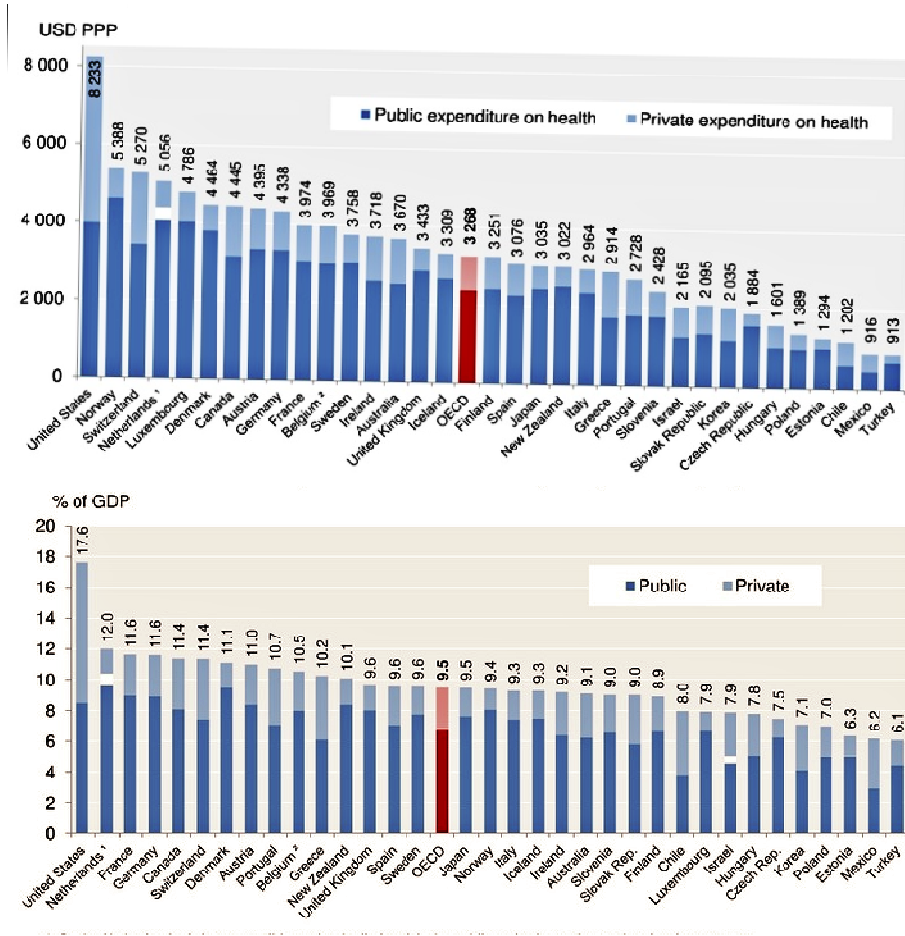
Looking at the quality of healthcare and its costs between countries gives interesting insights into the state of global healthcare. The US spends \$8,233 per person/year [17] which is 2.5-times more than most of developed nations and uses 17.6% of GDP for healthcare [17]. At the same time, the US had 2.4 practicing physicians per 1,000 people comparing to an average of 3.1 among OECD countries. In hospital beds per 1000 people, the US ranges with 2.6 well under the OECD average of 3.4.

Life expectancy in the US was increased by 9 years between 1960 and 2010; Japan's by 15 years and in OECD countries on average by 11 years [17]. In the drug development arena, per patient clinical trial costs have risen on average by 70 percent across all development phases since 2008 [18].

This numbers are clear indicators that the cost vs. outcome ratio needs to be improved [19] and the current models require adjustments

## 2.4 Data Ownership: Closed Data vs. Patient-Shared Access

Social media has arrived in healthcare. Patients are sharing publicly their own data, in which case no restrictions on scientific use apply. While many impediments by HIPAA compliance requirements to provide only selected, de-identified subsets to certain authorized individuals have been circumvented by such developments, new questions arise on the consequences from changes in data ownership and the shifts from hospital and providers to patient, and how this may impact integrated research.



**Fig. 1.** OECD Statistics 2012: Healthcare cost comparisons across countries (Source: OECD Health Data 2012 [2])

*Private and public expenditures per capita (upper panel) and in percentage of the GDP (lower pane)*

An article in 2011 in the Harvard Journal of Law and Technology opened the discussion about privacy concerns and why data ownership alone cannot resolve data access problems [20]. Nevertheless, with more and more internet connected personal health devices in use and a strong movement towards prevention and wellness, the aspect of data ownership on real-time, near-continuous monitoring of vital functions and their sharing among private individuals and physicians is a good indicator that closed data will lose ground against patient-shared access in the future.

## 3 Methodology for Change

### 3.1 Semantic Approach to Data Integration – Meaning, Inference, Reasoning

Resource description framework (RDF)-based integration (W3C standard) [21] opens new avenue and possibilities for rapid and efficient data integration. It has been around for quite some time, and its benefits as an agile, extensible environment built for interoperability have been widely demonstrated. Semantic data is much easier to connect, to visualize and extend as it does not rely on isolated RDBMS schemas, a contested standard data description, or a proprietary middleware translation layer. Dynamically built application ontologies can be adjusted and data remapped as needed; meaningful, not arbitrary schema-based connections drive its framework of triples, providing capabilities for inference and reasoning and pattern-based queries across the network graphs. The built-in basis for interoperability of true 5-star compliant RDF resources is not only a needed convenience, but a must for today's life sciences needs to utilize a fast array of publicly available linked data resources. RDF and its web ontology language, OWL [22] providing an excellent way to represent data with changing needs, with ability to reuse, repurpose in an easy to adopt and maintain fashion – allowing for disambiguous queries, pattern discovery and graph traversal across multiple RDF-represented resources.

In addition to being a globally standardized framework which links data based on their meaning, emergent properties include network visualization, visual query, faceted browsing, machine inference, and pattern recognition. Recent advances in provenance and versioning [23-24], in the development of public formal ontologies [25] and their direct accessibility through tools [26] have shown increasing interest in life sciences as foundation for larger project. Examples of such ongoing efforts are the development of the Translational Medical Ontology and Knowledgebase [27] driven by both, industry and academia, and the connex between medical informatics and bioinformatics in knowledge building in the clinic [28-29].

### 3.2 Linked Life Data, Linked Open Data – Consequences

The significant increase in the quality of Linked Data (LLD, LOD) [30-32] brings promising add-ons to qualify experimental findings early on through enrichment with external resources – but interoperability and different provenance remain still impediments for broader applicability as well as changes in licensing for previously 'open' public resources. Legal restrictions on use without modification prevent certain data resources from becoming interoperable as mapping and harmonization functions to other data cannot be applied.

As government funding for some linked open data cloud resources is unsure due to austerity and budget restraints in the US, Japan and Europe we will have to ensure to establish new business models between data provider and consumer to warrant continuous availability of such resources; either through private/academic/government partnerships or new concepts based on resource value for organizations. As the socio-economic benefits of maintaining these resources by far outweigh contributions towards their sustainability, such models will benefit all participants greatly.

### 3.3 Complexity and Change Require Dynamic, Adaptable Models

New and better scientific methods and analysis tools require adaptation for changes. As outlined before, the complexity of functional biology calls for network analysis of interconnected data in their relationships to each other. This leads logically to semantic integration approaches and network-driven systems to establish better understanding of complex biological intertwined reactions.

The healthcare industry is now about three years into '*meaningful use*', an ambitious incentive program to convince hospitals and private practices to use electronic health record (EHR) software. Regulators are also bringing HIPAA into the 21st century, and similar efforts are underway for telemedicine. Above all, it looks as if healthcare finally seems ready to benefit from big data, cloud services and other disruptive technologies that have dramatically changed other vertical industries [33]. As healthcare costs have tripled within the last decade and despite over \$10 Billion payments in healthcare incentives, we cannot afford to have EHR systems which are not interoperable and CRO's which are disconnected from their customers. A good example about possibilities in progressing with success in complex diseases like atherosclerosis to assess life threatening risk of plaque rupture via biomarkers leading to discovery of previously unknown pathway involvement, using such approaches to take advantage of integrated knowledge can be found in [34].

### 3.4 Understanding Biology: Shifting towards Interoperable, Integral Systems

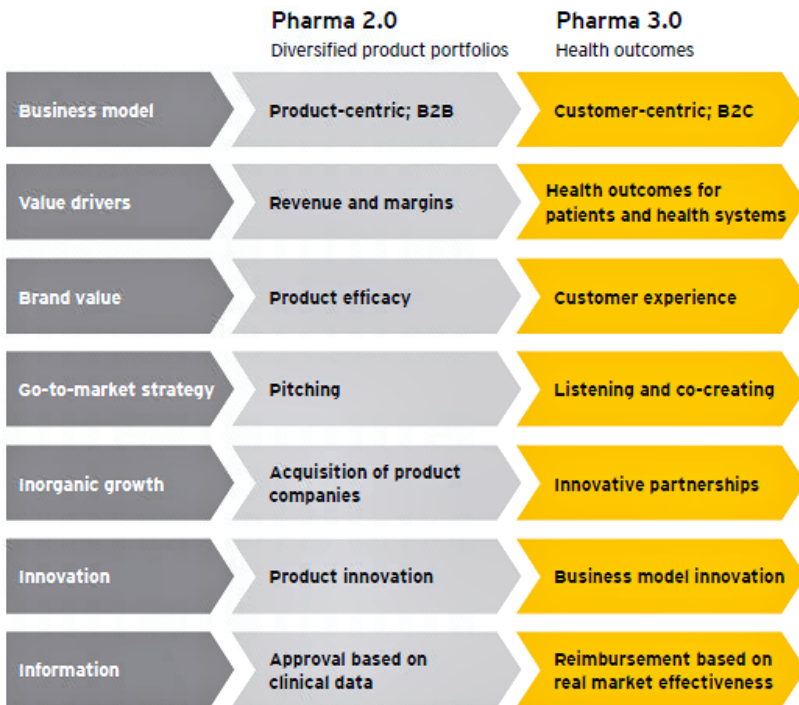
The need to contextualize experimental findings with pathway involvement and mechanisms is apparent as pharmacogenomics correlations not necessarily always match biological systems responses. Only when utilizing as much as we possibly can know, we will succeed in comparative effectiveness to select the best treatment at the right dose based on a patient's profile, lifestyle, disease stage and individual drug response.

The shift towards an integral view is the key to improving effectiveness of therapies and better understanding of the impact of a disease stage and a patient's profile on response, prognosis and outcome. Indications are that this is happening now. This year's Health Information and Managements Systems Society's (HIMSS13) conference brought the announcement, that five leading electronic health record (EHR) vendors were forming the '*CommonWell Health Alliance*' to promote 'seamless interoperability' of healthcare data [35].

### 3.5 Progression towards New Life Sciences Models: Pharma 3.0, Healthcare 3.0

There is a noticeable, albeit slowly, but steadily happening shift in industry from a product-centric business model to a customer/patient centric business model; and the new drivers are health outcomes. Maintaining or regaining growth will require the transition of Pharma from acquisition model to innovate partnerships and collaborative data sharing [36]. Innovation needs to focus more on business model

innovation than product innovation. Reimbursement needs to have its foundation in real market effectiveness rather than the approval of clinical trial data. What applies to Pharma, applies in equal ways to the entire healthcare industry. Efforts to reimbursement based on comparative effectiveness of clinical procedures and treatments indicate that life sciences industries are shifting to evidence-based and outcomes-focused business models – data-savvy, integrative, consumer (patient)-minded rather than product-centric. Moving quickly and following the value to progress towards a new sustainability model will be the key to success. Although there is a sense of urgency to try disruptive methods, it so far has been a ‘trying the water’ approach around edges of the business, not deeply embraced change. While the trends are apparent to most industry players, to think in new ways has always been uncomfortable and therefore slow in execution. If moving from 2.0 to 3.0 means, that collective impact approaches allow to move more expressively to pre-competitive sharing within the healthcare / life sciences space, the transition will not only be more rapid, it also will create new incentives for holistic approaches to this sector.



**Fig. 2.** Pharma on the move from 2.0 to 3.0 – Consequences for Life Sciences(Source: Ernst & Young 2012 [36])

*Significant changes need to occur in business model, value drivers and innovation during the shift towards a patient-centric, outcomes-focused and innovation-driven partnership model, where any reimbursement is based on effectiveness in its application rather than the product itself.*



## 4 Use Cases of Adoption

### 4.1 Pharmaceutical Industry

#### **Impact of Excipient Choice on Formulation Stability, Purity and Drug Efficacy**

Objective at a large Pharma company was the integration of disconnected chromatography data systems (CDS) and LIMS with compound and formulation databases to provide quick searches for compound purity data upon FDA inquiries. As there were no common identifiers, such inquiries required time-consuming off-line searches with ambiguous results, delaying responses to the FDA by several weeks. A system was sought to remedy this organizational problem.

The solution to build a semantic platform for compound purity and stability assessment not only was accomplished in a fraction of the allocated time (6 weeks instead of ~ 1 yr. projected project completion time in traditional data warehouse fashion), but inference could be used to quickly and unambiguously find the desired raw data in the CDS. As a pleasant side effect of the semantic data model implemented, an additional data resource was integrated to allow for queries determining the impact of the choice of excipient in a given drug formulation on active ingredient efficacy and overall drug stability.

#### **Pre-clinical Toxicity Assessment and Compound Toxicity Type Classification**

In a joint project (NIST/Cogenics/CLDA) to understand the impact of toxicity on biological systems, sets of known and presumed toxicants were used in large 3-year animal studies to determine biomarker classifier patterns and their applicable ranges for pre-clinical toxicity screening of compounds.

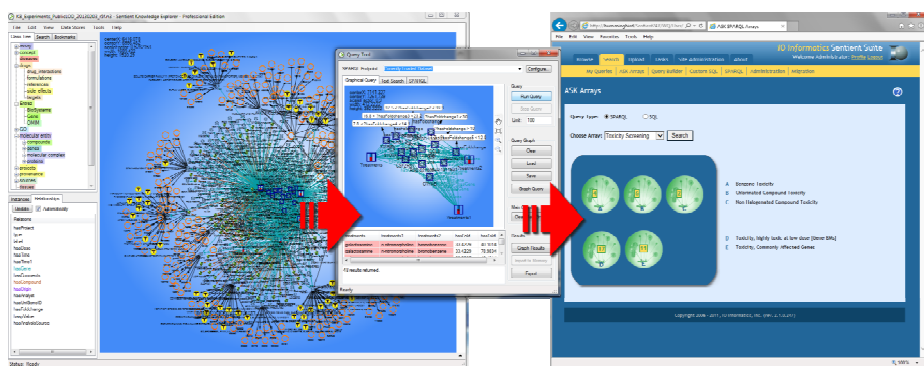
Hepatotoxicity studies consisted of a panel of hepatotoxicants at single oral dose (placebo, low, mid, high) in groups of 4 rats, at 6, 24 and 48 hrs.) and metabolic analysis of liver, serum and urine (1603 metabolic components; Bruker LC/MS-MS); gene expression microarray analysis of liver and whole blood (31096 transcript probes; Affymetrix); and statistical biomarker pre-selection at  $p < 0.005$ ,  $\text{abs } fc > 10$  (genes) and  $p < 0.005$ ,  $\text{abs } fc > 2.5$  (metabolites).

Alcohol studies were carried out at high doses t.i.d. for four days, with and without 24h withdrawal; metabolic analysis of plasma, liver and brain (1620 metabolic components), microarray analysis of liver and brain (31096 transcript probes) and statistical biomarker pre-selection at  $p < 0.005$ ,  $\text{abs } fc > 5$  (genes) and  $p < 0.005$ ,  $\text{abs } fc > 2.5$  in similar fashion.

The experimental network of statistically preselected putative genomic and metabolomic biomarkers was then enriched with public RDF resources through SPARQL queries to discover common pathway dependencies, using LOD-based systems-biological qualification of experimental pharmacogenomic correlations. As a result of semantic data integration, markers to distinguish several distinct types of

Table 2. Toxicity biomarker with biological validation: Characterization of toxicity types

BM Type	Instance	UniProt AC	Pathway Gene	Protein	Biology
gene	CYP2C40	P11610	cd3cc	Cytochrome P450 2C40	heme binding, iron on binding, aromatase activity
gene	AKR7A3	P33918	acr7a3	Alcatorin B1 alcohol dehydrogenase member 3	decarboxylation
gene	GPX2	P83845	gpx2	Glutathione peroxidase 2	response to oxidative stress, negative regulation of inflammatory response
gene	MYC	P09416	myc	Myb proto-oncogene protein (Transcription factor p64)	regulation of gene transcription, non-specific DNA binding, activates transcription of growth-related genes
gene	MT1A	P02803	mt1a	Metallothionein-1	metal ion binding
gene	HMOX1	P03782	hmox1	Heme oxygenase-1	heme catabolic process, negative regulation of DNA binding
gene	FGF21	C91080	fgf21	Fibroblast growth factor 21 (Protein Fgf21)	positive regulation of ERK1 and ERK2 cascade, MAPKXX cascade and cell proliferation
gene	AKR1B6	C91W93	acr1b6	Aldose reductase-like protein	oxidoreductase activity
gene	TRB3	C91W706	trb3	Tribbles homolog 3	disrupts insulin signaling by binding directly to Akt kinases, expression induced during programmed cell death
gene	YC2	P49418	gsta5	Glutathione S-transferase alpha-5 (EC 2.5.1.18)	response to drug, xenobiotic catabolic process
gene	ABCB1, RGD:619651	P43245	abcb1	Multidrug resistance protein 1 (EC 3.6.3.44)	response to organic cyclic compound, tumor necrosis factor, a semi-containing substance or ionizing radiation
gene	RGD:1370691	G912F3	Zrand2a	ANK-type zinc finger protein 2A	zinc ion binding
gene	GSTP1, GSTP2	P04906	gstp1	Glutathione S-transferase P (EC 2.5.1.18)	response to toxin, xenobiotic metabolic process, response to reactive oxygen species, response to ethanol
gene	RGD:706417	C82789	ugl2p7	UDP-glucuronosyltransferase 2B7 (UDP-GT 2B7) (EC 2.4.1.17)	major importance in conjugation and subsequent elimination of toxic xenobiotics and endogenous compounds
gene	GCLC	P19468	gclc	Glutamate-cysteine ligase catalytic subunit (EC 6.3.2.2)	response to oxidative stress
gene	TNRD1	C89049	tnrd1	Thioredoxin reductase 1, cytoplasmic (EC 1.8.1.9)	benzene-containing compound metabolic process, cell redox homeostasis, response to drug
gene	MG01	P03962	mgp1	NAD(P)H dehydrogenase (quinone) 1 (EC 1.6.6.2)	response to oxidative stress, response to ethanol, superoxide dismutase activity
gene	DDIT4L	C870E0	ddit4l	DNA damage-inducible transcription factor 4-like protein	negative regulation of signal transduction, inhibits cell growth by regulating TOR signaling pathway
metabolite	Pyrogallanic acid	C9ER34	aco2	Aconitase hydrolase, mitochondrial	citrate metabolism, isocitrate metabolism, tricarboic acid cycle
metabolite	Choline	C84C57	cdit1a1	Choline-aminotransferase, s-adenosylmethionine-dependent (EC 1.2.1.31)	betaine biosynthesis via choline pathway, response to DNA damage stimulus



**Fig. 3.** Use case: Pre-clinical toxicity assessment and categorization

*Network view of toxic insult from a set of compounds with affected genes and metabolites in conjunction with their associated pathways and diseases (left); visual SPARQL pattern query with ranges for each biomarker (center); published query pattern as Applied Knowledgebase (ASK) arrays, accessible on a web server for rapid toxicity type screening of compounds (right)*

toxicity were established, which passed functional biology criteria for toxicity. As a result, Benzene-, Chlorinated compound- and Ethanol toxicity could be distinguished and long-term effects of Alcohol on memory functions in brain biologically recognized [37-38]. The findings from these studies have major implications for pre-clinical toxicity assessment and were reported at a FDA workshop on pharmacological mechanism-based drug safety assessment and prediction in 2011 [39]. Results for Benzene-type toxicity are depicted in Table 2

### **Integral Systems View on Drug Safety and Adverse Effects**

A multi-national Pharma project required data integration of trial management for drug safety assessment from a large document corpus. The project scope involved ~14,000 documents (~8 gb) with ~1.2 billion spreadsheet cells containing data. Data quality was unknown and data curation requirements were not obvious at start.

During semantic integration, thesaurus-based harmonization and transformation during mapping and data quality enhancement through inference, inconsistencies in the body of data were detected and remedied, As result, the initial semantic knowledgebase contained ~ 780 M triples. Parameterizable SPARQL and a drag-&-drop query builder using a cache search index (iPool) for fast queries provide easy access through a web portal for integral adverse effect queries on compounds used in clinical trials.

## 4.2 Government

### Cross-Species Biomarkers to Reduce Animal Studies

In an effort to reduce animal testing in accordance with world-wide trends against animal experiments, the FDA Center for Veterinarian Medicine (CVM) started a long-term project to develop species-independent biomarkers. The project involved the need to integrate genomics data, proteomics data, imaging endpoints from biopsies, assay results and animal data obtained from a variety of species to proof feasibility to determine common disease biomarkers across those species.

Objective was exploring the ability to move from large animal testing (pigs, dogs) to smaller ones (rats, mice), and further progress to human cell cultures in an effort to reduce costs and minimize the need for animal experiments.

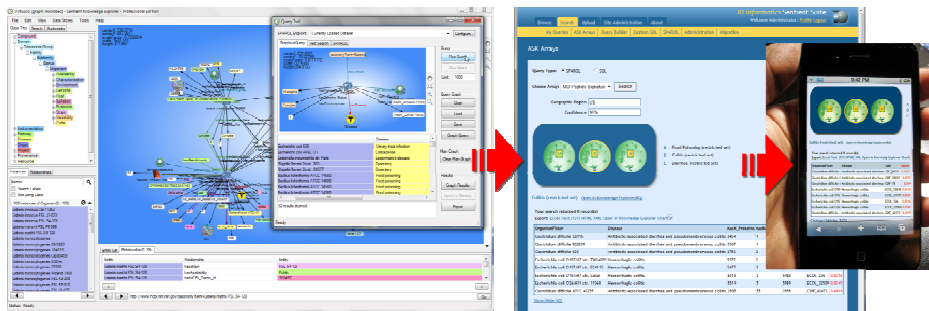
The semantic integration of raw data and results from experimental tests, images, and animal characteristics across multiple time points and the incorporation of public resources (UniProt, KEGG, Reactome) into a comprehensive multi-species biological knowledgebase provides the basis for network analyses to discover cross-species biomarkers applicable to human adverse events and diseases.

Initial results have been very encouraging, and ongoing research integrating additional species data is under way. Ultimately, this development will lead to a significant reduction in animal testing and better drug models for human responses.

### Microbial Pathogen Knowledgebase to Identify Biological Threads

Identification of biological threads or the outbreak characteristics of infectious diseases require rapid action on proper identification and characterization of the biological source. While there are several public database resources available (ICTV, MIST, PATRIC), their schemas are not built for interoperability.

As the need to identify microbial pathogens quickly and precisely entails the integral access to as many resources as possible, a semantic mapping for those public resources (including a thesaurus for microorganism for synonym harmonization) was



**Fig. 4.** Use case: Quick pathogen identification from samples using microbial knowledgebase

*Visual SPARQL queries across the graph of an integrated microbial pathogen knowledgebase are used to identify threads from experimental data. Those pattern are populating a web server's ASK arrays for online and in-field screening using smart phones*

established and the microbial pathogen knowledgebase enhanced with NCBI's taxonomy for group classification into family, subfamily, genus and organisms. This Knowledgebase [40-42] then was used in integrated network analysis of samples from MS-based sequencing and/or rapid microbiological assays in conjunction with historic case data to identify pathogens within the host samples.

An Applied Semantic Knowledgebase (ASK) web portal for simplified pattern queries was established to determine threads by location and confidence level for the identified pathogens.

### 4.3 Clinical Decision Support

#### **Biomarkers in Transplantation: Organ Rejection Screening**

Transplantation is currently the most common therapy for patients with end-stage organ failure. It involves putting a donated organ into the immunologically foreign environment of the receiving patient. While the transplantation procedure itself may go smoothly, the recipient's immune system may react to the new organ to induce rejection. White blood cells and antibodies are primarily involved in the recognition, attack and destruction of foreign tissues, yielding dysfunction of the transplanted organ.

A major challenge facing clinical caregivers in the management of organ rejection is to determine whether a transplanted organ is undergoing rejection prior to any symptoms. This typically required using highly invasive and risky procedures, such as tissue biopsies – expensive, regularly performed emotionally and physically stressful procedures which may still result in inconclusive findings. In order to prevent organ rejection, powerful therapies are used to suppress a patient's immune system. While this approach reduces the probability of rejection, it does so at a high cost. Impairment of a recipient patient's immune system leaves them susceptible to infections, malignancies and functional complications in the newly transplanted organs.

As individuals vary in their response to such therapies, understanding this variation would help physicians balancing the necessity of therapy with its possible side-effects. The ability to personalize immune suppressants for each patient not only alleviates patient discomfort and side-effects, but also reduces the enormous costs associated with over-prescription of immunosuppressive drugs and other diagnostic procedures.

The Biomarkers in Transplantation (BIT) initiative was established to identify and validate biomarkers for diagnosis of rejection of a transplanted organ via a simple blood test [43]. The program was launched in 2004 to better understand acute or chronic tissue rejection in heart, liver, and kidney transplant patients. Its application to use a web-based Applied Knowledgebase (ASK) decision support system won Bio-IT's Best Practices Award in 2010 [44]. It utilizes semantic data integration and parameterized SPARQL queries with weighing and ranges for multimodal biomarkers [45-46] to provide screening for patients at risk

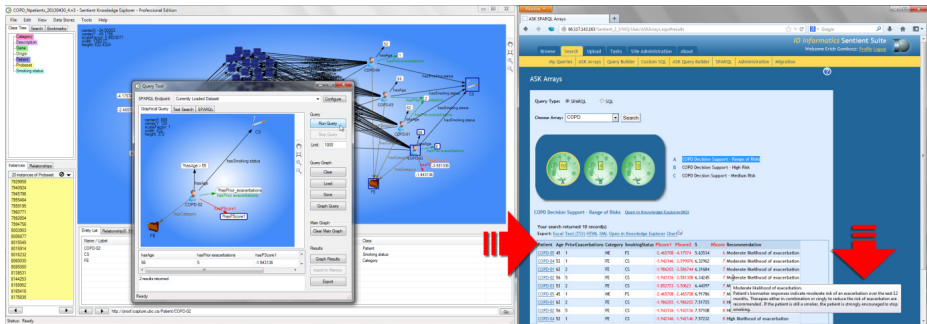


**Fig. 5.** Use case: Score-based recommendation for immune suppression therapy at risk of organ rejection

*Semantic network of a heart transplant patient with blood-test based biomarkers, disease history and organ donor information (left). Use of ASK array to screen for rejection risk and provide guidance-text based decision support on immune suppression therapy (right).*

**Biomarkers for COPD: Prediction of Exacerbation**

Chronic obstructive pulmonary disease (COPD) is a chronic, progressive disease characterized by loss of lung function and breathlessness that reduce quality of life, productivity, and longevity. Its course is frequently complicated by acute exacerbations related to respiratory infections, ambient pollution or poor management of disease. Results are urgent visits to physicians, emergency room care, hospitalization, intensive care unit admissions, and even death. COPD is a major cause of morbidity and mortality around the world. In British Columbia (BC) and the



**Fig. 6.** Use case: Biomarker-based clinical decision support system to predict COPD exacerbations

*Semantic integration of clinical data for genomic marker-based decision support predicting likelihood of exacerbation. Network view of 5 patients with different smoking history and prior exacerbations (left). ASK array physician guidance based on normalized algorithmic scoring of weighed biomarker expression profiles (right)*

rest of Canada, COPD exacerbations are leading cause of hospitalization, and prevalence has continued to rise, increasing 41% since 1982. It now affects 10-14% of Canadians 40 years of age and older and 600 million people globally [47-48]. As the 4th leading cause of mortality in Canada and the US [49] and the only major cause of mortality for which death rates continue to rise [48-49], the economic costs of COPD management for society are estimated to be over a billion dollars annually in Canada with \$736 million of that directly attributable to exacerbations [51]. Most of the morbidity occurs during exacerbations, and their direct costs are predicted to surpass \$1 billion by 2015 [52].

A biomarker-based decision support system to predict likelihood of exacerbation and advise treating physician via web-based access to screen patients alleviates those risks and provides a tremendous improvement for patient care in reducing emergency care and hospitalization.

## **5 Discussion, Future Outlook**

### **5.1 Applied Knowledge as Cost Saver**

A 2012 released OECD Health Data Report [19] provides among others statistics on health expenditures, healthcare utilization, demographic references, healthcare quality indicators, pharmaceutical market and long-term care resources. In 2010, in the US public expenditure on health were 48.2% from the total expenditures compared to 87.7% in the Netherland, 85.1% in Denmark and 83.2% in the UK [19]. Applied knowledge from semantic integration of experimental, clinical and public proteomics, genomics, metabolomics and pathway resources led to the development and qualification of multi-modal biomarker pattern applicable for rejection risk assessment with enormous cost savings.

Taking the examples from 4.3 using biomarker blood test for clinical decision support can be used replacing monthly biopsies for up to a year after transplantation at average costs of \$4000.-/each. As of 2007, the average price of a kidney-only transplant was \$246,000 in the first year. A single lung transplant totaled \$399,000. A heart transplant patient's first-year total medical costs were \$658,000. In 2011, in the US 1,760 patients on the heart wait list received heart transplants. This represents a decrease from 2,333 hearts transplanted in 2010 and 2,211 in 2009 (Source: UNOS/OPDN). The Heart and Stroke Foundation in Canada reports that heart disease and stroke costs the Canadian economy more than \$20.9 billion every year in physician services, hospital costs, lost wages and decreased productivity (Conference Board of Canada, 2010). In 2010, there were 167 heart transplants in Canada, with 135 patients on the waiting list for organ donors. One can imagine the cost savings and quality of life enhancement for patients obtainable through widespread use of preventive non-invasive screening methods. Similarly, the effects of being able to predict COPD exacerbations which cause permanent lung tissue damage, are impressive indicators how far reaching integral patient-centric procedures based on semantic knowledgebases have influenced the socio-economics of healthcare.

Actionable knowledge and near-real time alerting of physicians about patients at risk in life-threatening conditions is a testimonial to the real value of interoperable, agile data in life sciences.

## 5.2 Socio-economics - Higher Quality of Life

As exemplified by examples provided, the use of semantic data integration technologies and integrated, harmonized network approaches utilizing both, internal experimental, clinical, observational and demographic data and public resources provided as RDF/OWL via SPARQL endpoints to enrich, qualify, validate – even plan additional new experiments – has moved from exploratory projects to mainstream acceptability.

Biomarker-based screening for kidney disease to avoid biweekly dialysis or transplantation, heart organ transplant monitoring with biomarkers instead of costly and unpleasant monthly biopsies, and prediction of exacerbations in COPD are just the beginning of a new era of patient-centric, data-driven improvement in health outcomes where everyone involved in applying Pharma 3.0 and Healthcare 3.0 principles [36] will win back sustainability based on reimbursement of real, not perceived effectiveness at the reward of huge socio-economic benefits and improved prevention, care and quality of life.

## 5.3 Actions Today and Tomorrow

We can see already today the adaption towards more open-minded strategic approaches to build integrated, interoperable (and open?) life science knowledge system capable of remarkable results at significantly lower costs [53] – but there still remains a lot to do.

We need to do more to promote and proliferate these efforts among wider communities to ensure that the life sciences industries are sustainable, effective and applied to help through early intervention, better prognosis and integrated patient-centric, knowledge-based treatment to improve outcomes, increase life expectancy and the quality of life for all. I would urge you to join me in my assessment, that we cannot afford to wait any longer, and that the phase of hesitation on early adaptation to implement innovate solutions and business processes in our quest for comprehensive, integrative systems approaches to better understand biology is over.

We know, what is necessary to change the model, and we have examples leading the way to a bright future – but knowing is not enough; it's time to act, and more than any time before, the time is now.

*“Knowing is not enough; we must apply. Willing is not enough; we must do.”*  
- Johann Wolfgang von Goethe (1782)

**Acknowledgements.** The following groups and researchers have been contributing to the success of the projects described in 4. Use Cases of Adoption:



*Toxicity Project:* Pat Hurban, Alan J. Higgins, Imran Shah, Hongkang Mei, Ed K. Lobenhofer (Cogenics, Morrisville, NC), Fulton T. Crews (Bowles Center for Alcohol Studies / UNC, Chapel Hill, NC)

*Microbial Pathogen Project:* Sherry Ayers (FDA NARMS, Silver Spring, MD)

*Species-independent Biomarkers:* Haile F. Yancy, Michael J. Myers, Rudell Screven (FDA VET / CVM, Laurel, MD)

*Biomakers in Transplantation and COPD:* Bruce Mc Manus, Raymond T. Ng, Scott Tebbutt (Centre for the Prevention of Organ Failures / PROOF, Vancouver, BC, Canada)

*RDF / OWL Database Resources and Ontologies:* Jerven T. Bolleman (Swiss Institute Bioinformatics / SIB / UniProt Consortium, Geneva, Switzerland), Michel Dumontier (Bio2RDF II, Carleton University, Ottawa, Canada), Mark A. Musen, Patricia L. Whetzel (BMIR / NCBO Stanford, CA)

*W3C HCLS LLD / Pharmacogenomics SIG:* Scott Marshall, Michel Dumontier

*IO Informatics:* Andrea Splendiani, Jason A. Eshleman, Robert A. Stanley

*Working Groups:* Best Practices in Data Sharing, Informatics for Personalized Medicine

*Grant Support for Toxicity Studies:* NIST ATP #70NANB2H3009, NIAAA #HHSN281200510008C

## References

1. Technology survey on LIMS and ELN in Life Sciences. Project Share Biotech, University of Nantes, France (2011), [http://www.biogenouest.org/sites/default/files/Biogenouest/Fichiers/qualite/lims\\_eln\\_sharebiotech\\_study\\_report.pdf](http://www.biogenouest.org/sites/default/files/Biogenouest/Fichiers/qualite/lims_eln_sharebiotech_study_report.pdf)
2. OECD health systems 2012 (2012), <http://www.oecd.org/health/health-systems/oecdhealthdata2012.htm>
3. Richter, B.G., Sexton, D.P.: Managing and Analyzing Next-Generation Sequence Data. *PLoS Comput. Biol.* 5(6), e1000369 (2009), doi:10.1371/journal.pcbi.1000369, <http://www.ploscompbiol.org/article/info%3Adoi%2F10.1371%2Fjournal.pcbi.1000369#s5>
4. Wearable wireless health sensor for remote bio-monitoring. *Gizmag* (2010)
5. Sensors facilitate health monitoring. *Sensor Magazine* (2011), <http://www.sensormag.com/specialty-markets/medical/sensors-facilitate-health-monitoring-8365>
6. 10 wearable health tech devices to watch. *Information Week* (2012), <http://www.informationweek.com/healthcare/mobile-wireless/10-wearable-health-tech-devices-to-watch/240012613>
7. Mobile ECG monitor via iPhone. *AliveCor* (2011), <http://www.alivecor.com/?gclid=CPP9yqyL-LUCFchaMgodg3QANA>
8. fitbit one – Wireless activity monitor. *Fitbit* (2011), <http://www.fitbit.com/one/gallery>
9. Sanjeev, A., Boaz, B.: *Computational Complexity – A Modern Approach*, Cambridge (2009) ISBN 978-0-521-42426-4
10. Zax, D.: *MIT Technology Review*, November 2011: Is Personal Data the New Currency? (2011), <http://www.technologyreview.com/view/426235/is-personal-data-the-new-currency/>

11. Ohlhorst, F.J.: *Big Data Analytics: Turning Big Data into Big Money*. Wiley (2012) ISBN: 978-1-118-14759-7, <http://www.wiley.com/WileyCDA/WileyTitle/productCd-1118147596.html>
12. Dean, M.: Crowdsourcing as Healthcare Policy Pros&Cons. *The Nation* (2013), <http://www.thenation.com/blog/174059/crowdsourcing-health-care-policy-cant-we-do-better>
13. Engelen, L.: *Crowdsource your health*. Video on TED.com (2012), [http://www.ted.com/talks/lucien\\_engelen\\_crowdsource\\_your\\_health.html](http://www.ted.com/talks/lucien_engelen_crowdsource_your_health.html)
14. Davis, K.: *The \$1,000 Genome. The Revolution in DNA Sequencing and the New Era of Personalized Medicine*. Free Press (2010) ISBN 9781416569596
15. Mardis, E.: The \$1,000 genome, the \$100,000 analysis? *Genome Medicine* 2, 84 (2010)
16. Herper, M.: The truly staggering costs of inventing new drugs. *Forbes* (2012), <http://www.forbes.com/sites/matthewherper/2012/02/10/the-truly-staggering-cost-of-inventing-new-drugs/>
17. PBS, 2012: *Healthcare costs: How the US compares with other countries* (2012), <http://www.pbs.org/newshour/rundown/2012/10/health-costs-how-the-us-compares-with-other-countries.html>
18. Evangelista, E.: Per patient clinical trial costs rise 70% in three years. *Marketwire* (2011), <http://www.marketwire.com/press-release/per-patient-clinical-trial-costs-rise-70-in-three-years-1538269.htm>
19. OECD Health Statistics: *Data Dissemination and Results* (2012), [http://www.oecd.org/els/health-systems/Item%202\\_OECD%20Health%20Data\\_Dissemination%20and%20Results\\_MCC\\_Final.pdf](http://www.oecd.org/els/health-systems/Item%202_OECD%20Health%20Data_Dissemination%20and%20Results_MCC_Final.pdf)
20. *Harvard Journal of Law and Technology*: 25(1) 2011: Much Ado about Data Ownership (2011), <http://jolt.law.harvard.edu/articles/pdf/v25/25HarvJLTech69.pdf>
21. *Resource Description Framework (RDF): Concepts and Abstract Syntax* (2004) W3C Recommendation (February 10, 2004), <http://www.w3.org/TR/rdf-concepts/>
22. *OWL 2 Web Ontology Language - Document Overview* (2nd edn.) (2012) W3C Recommendation (December 11, 2012), <http://www.w3.org/TR/owl2-overview/>
23. Cyganiak, R., Zhao, J., Alexander, K., Hausenblas, M.: *VoID Vocabulary of Interlinked Datasets*. DERI, W3C note (2011)
24. *PROV-O: The PROV Ontology*. W3C Candidate Recommendation (2012)
25. Musen, M.A., Noy, N.F., Shah, N.H., Whetzel, P.L., Chute, C.G., Story, M.A., Smith, B.: The National Center for Biomedical Ontology. *J. Am. Med. Inform. Assoc.* 19(2), 190–195 (2012)
26. Salvadores, M., Horridge, M., Alexander, P.R., Ferguson, R.W., Musen, M.A., Noy, N.F.: Using SPARQL to Query BioPortal Ontologies and Metadata. In: Cudré-Mauroux, P., et al. (eds.) *ISWC 2012, Part II*. LNCS, vol. 7650, pp. 180–195. Springer, Heidelberg (2012)
27. Luciano, J.S., Andersson, B., Batchelor, C., Bodenreider, O., Clark, T., Denney, C.K., Domarew, C., Gambet, T., Harland, L., Jentzsch, A., Kashyap, V., Kos, P., Kozlovsky, J., Lebo, T., Marshall, S.M., McCusker, J.P., McGuinness, D.L., Ogbuji, C., Pichler, E., Powers, R.L., Prud'hommeaux, E., Samwald, M., Schriml, L., Tonellato, P.J., Whetzel, P.L., Zhao, J., Stephens, S., Dumontier, M.: *The Translational Medicine Ontology and Knowledge Base: driving personalized medicine by bridging the gap between bench and bedside*. *J. Biomed. Semantics* 2(suppl. 2), S1 (2011)

28. Connecting medical informatics and bioinformatics: Advances in knowledge acquisition and management. Mayo Clinic's Enterprise Data Trust. JAMIA (2010), <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3000789/>
29. Podgorelec, V., Grašič, B., Pavlič, L.: Medical diagnostic process optimization through the semantic integration of data resources: New opportunities for automated reasoning at patient-data level: Medical diagnostic process optimization through the semantic integration of data resources. *Computer Methods and Programs in Biomedicine* (2009), <http://www.sciencedirect.com/science/article/pii/S0169260709000832>
30. Linked Life Data <http://linkedlifedata.com/sources> LLD: Instance mapping, Predicate mapping; 1.1: no of statements: 8,740,201,002;EU IST project
31. Linked Open Data <http://linkeddata.org/> LOD Cloud Diagram: <http://lod-cloud.net/> LDOW2012 Linked Data on the Web. Bizer, C., Heath, T., Berners-Lee, T., Hausenblas, M.: WWW Workshop on Linked Data on the Web, Lyon, France (April 16, 2012)
32. Callahan, A., Cruz-Toledo, J., Ansell, P., Klassen, D., Tumarello, G., Dumontier, M.: Improved dataset coverage and interoperability with Bio2RDF Release 2. In: SWAT4LS Workshop, Paris, France, November 30 (2012)
33. 13 healthcare IT trends in 2013, <http://www.cio.com/slideshow/detail/83055>
34. Plasterer, T.N., Stanley, R., Gombocz, E.: Correlation Network Analysis and Knowledge Integration. In: Dehmer, M., Emmert-Streib, F., Graber, A., Salvador, A. (eds.) *Applied Statistics for Network Biology: Methods in Systems Biology*. Wiley-VCH, Weinheim (2011) ISBN: 978-3-527-32750-8
35. Is healthcare IT interoperability (almost) here? HIMSS13, [http://www.cio.com/article/731757/Is\\_Healthcare\\_IT\\_Interoperability\\_Almost\\_Here\\_](http://www.cio.com/article/731757/Is_Healthcare_IT_Interoperability_Almost_Here_)
36. Ernst & Young: The third place: healthcare everywhere – Global Life Sciences Report (2012), [http://www.ey.com/Publication/vwLUAssets/Progressions\\_The\\_third\\_place\\_health\\_care\\_everywhere\\_-\\_Global\\_Life\\_Sciences\\_Report\\_2012/\\$FILE/Progressions\\_Global\\_Life\\_Sciences\\_Report\\_2012\\_The\\_third\\_place\\_health\\_care\\_everywhere.PDF](http://www.ey.com/Publication/vwLUAssets/Progressions_The_third_place_health_care_everywhere_-_Global_Life_Sciences_Report_2012/$FILE/Progressions_Global_Life_Sciences_Report_2012_The_third_place_health_care_everywhere.PDF)
37. Gombocz, E.A., Higgins, A.J., Hurban, P., Lobenhofer, E.K., Crews, F.T., Stanley, R.A., Rockey, C., Nishimura, T.: Does network analysis of integrated data help understanding how alcohol affects biological functions? - Results of a semantic approach to biomarker discovery. 2008 Biomarker Discovery Summit 2008, Philadelphia, PA (2008)
38. Higgins, A.J., Gombocz, E.A., Stanley, R.A.: From Correlation to Biological Understanding: Multi-modal semantic networks for biomarker discovery and qualification. In: Bio-IT World 2008, Boston, MA (2008), [http://www.io-informatics.com/news/pdfs/CHI\\_BioIT2008\\_Talk.pdf](http://www.io-informatics.com/news/pdfs/CHI_BioIT2008_Talk.pdf)
39. Gombocz, E., Stanley, R.: Predictive Toxicology: Applied Semantics with a major impact on drug safety. In: FDA Drug Safety Workshop: Pharmacological Mechanism-Based Drug Safety Assessment and Prediction FDA, White Oak Campus, Silver Spring, MD (2011)
40. Gombocz, E., Candlin, J.: A Novel Approach to Recognize Peptide Functions in Microorganisms: Establishing Systems Biology-based Relationship Networks to Better Understand Disease Causes and Prevention. In: 8th Annual Conference US Human Proteome Organization: The Future of Proteomics (HUPO 2012), San Francisco, CA (2012)

41. Gombocz, E., Candlin, J.: How semantic technology helps fighting infectious diseases: Biological systems approach to understand microbial pathogens. In: *Semantic Technology & Business Conference (SemTech 2012)*, San Francisco (2012), [http://www.io-informatics.com/news/pdfs/SemTech2012\\_EGombocz\\_Talk20120606\\_r3.pdf](http://www.io-informatics.com/news/pdfs/SemTech2012_EGombocz_Talk20120606_r3.pdf)
42. Gombocz, E., Candlin, J., Stanley, R., Chiang, D.: *Semantically Enhancing Protein Identification: Systems Biology Knowledgebase for Infectious Disease Screening*. In: *Bio-IT World 2012*, Boston, MA (2012), [http://www.io-informatics.com/news/pdfs/BioIT2012\\_Poster.pdf](http://www.io-informatics.com/news/pdfs/BioIT2012_Poster.pdf)
43. *A new way to predict and diagnose organ rejection*. Transplant Foundation Research of British Columbia (2010), [http://www.trfbc.org/site/PageServer?pagename=News\\_Biomarkers](http://www.trfbc.org/site/PageServer?pagename=News_Biomarkers)
44. *Bio-IT World Best Practices Award 2010 in Personalized & Translational Medicine: PROOF / iCAPTURE Centre of Excellence; Semantic Data Integration, Knowledge Building and Sharing Applied to Biomarker Discovery and Patient Screening for Pre-symptomatic Heart, Lung or Kidney Failure in Transplantation Medicine* (2010), <http://www.prweb.com/releases/2010/04/prweb3917414.htm>
45. Ng, R.T., Gombocz, E.: *Biomarker Development to Improve Decision Support for the Treatment of Organ Failures: How Far Are We Today?* In: *ADAPT 2010*, Arlington, VA (2010), [http://www.io-informatics.com/news/pdfs/ADAPT2010\\_Talk.pdf](http://www.io-informatics.com/news/pdfs/ADAPT2010_Talk.pdf)
46. Stanley, R., McManus, B., Ng, R., Gombocz, E., Eshleman, J., Rockey, C.: *W3C Semantic Web Use Cases and Case Studies Case Study: Applied Semantic Knowledgebase for Detection of Patients at Risk of Organ Failure through Immune Rejection*. Joint Case Study of IO Informatics and University British Columbia (UBC), NCE CECR PROOF Centre of Excellence, James Hogg iCAPTURE Centre, Vancouver, BC, Canada (2011)
47. *Global Initiative for Chronic Obstructive Lung Disease, Global Strategy for the Diagnosis, Management, and Prevention of Chronic Obstructive Pulmonary Disease* (2009) (updated), <http://www.goldcopd.com/Guidelineitem.asp?l1=2&l2=1&intId=2003>
48. Murray, C.J., Lopez, A.D.: *Alternative projections of mortality and disability by cause 1990-2020: Global Burden of Disease Study*. *Lancet*. 349(9064), 1498–1504 (1997)
49. Jemal, A., Ward, E., Hao, Y., Thun, M.: *Trends in the leading causes of death in the United States, 1970-2002*. *JAMA* 294(10), 1255–1259 (2005)
50. Buist, A.S., McBurnie, M.A., Vollmer, W.M.: *International variation in the prevalence of COPD (the BOLD Study): a population-based prevalence study*. *Lancet*. 370(9589), 741–750 (2007)
51. Mathers, C.D., Loncar, D.: *Projections of global mortality and burden of disease from 2002 to 2030*. *PLoS Med.* 3(11), e442 (2006)
52. Mittmann, N., Kuramoto, L., Seung, S.J., Haddon, J.M., Bradley-Kennedy, C., Fitzgerald, J.M.: *The cost of moderate and severe COPD exacerbations to the Canadian healthcare system*. *Respir. Med.* 102(3), 413–421 (2008)
53. Gombocz, E.A.: *On the road to production: Semantic integration cases indicate successful adoption to improve knowledge-based decisions in Pharma and healthcare* NCBO Webinar Series, Stanford, CA. Recording (2013)

# Ibidas: Querying Flexible Data Structures to Explore Heterogeneous Bioinformatics Data

Marc Hulsman<sup>1</sup>, Jan J. Bot<sup>1</sup>, Arjen P. de Vries<sup>1,3</sup>, and Marcel J.T. Reinders<sup>1,2</sup>

<sup>1</sup> Delft Bioinformatics Lab, Delft University of Technology

<sup>2</sup> Netherlands Bioinformatics Centre (NBIC)

<sup>3</sup> Centrum Wiskunde & Informatica (CWI)

**Abstract.** Nowadays, bioinformatics requires the handling of large and diverse datasets. Analyzing this data demands often significant custom scripting, as reuse of code is limited due to differences in input/output formats between both data sources and algorithms. This recurring need to write data-handling code significantly hinders fast data exploration.

We argue that this problem cannot be solved by just data integration and standardization alone. We propose that the integration-analysis chain misses a link: a query solution which can operate on diversely structured data throughout the whole bioinformatics workflow, rather than just on data available in the data sources. We describe how a simple concept (shared 'dimensions') allows such a query language to be constructed, enabling it to handle flat, nested and multi-dimensional data. Due to this, one can operate in a unified way on the outputs of algorithms and the contents of files and databases, directly structuring the data in a format suitable for further analysis. These ideas have been implemented in a prototype system called Ibidas. To retain flexibility, it is directly integrated into a scripting language. We show how this framework enables the reuse of common data operations in different problem settings, and for different data interfaces, thereby speeding up data exploration.

## 1 Introduction

Research in the field of biological systems has become a strongly data-driven activity. Measurements are performed at multiple levels (genomics, transcriptomics, etc.), and combined with already-available information, which can be accessed through the more than 1300 available public data sources [1]. Handling these large and diverse datasets can be a time-consuming and complex task, requiring the development of many custom-written data-handling scripts. This problem has attracted significant attention from researchers, which has led to the development of numerous approaches to improve this process [2].

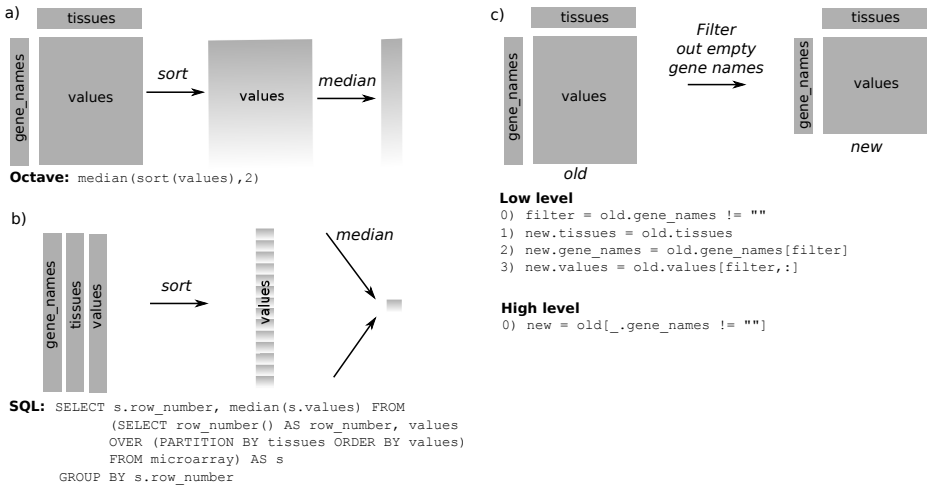
Generally this is solved in a bottom-up fashion, where one starts from the data sources and enables structured access to the data, for example by making use of warehouses, webservices or the semantic web, e.g. [3–5]. Bottom-up approaches have some limitations however. Data interfaces offer relatively limited functionality (for computational and security reasons as queries often run on

public servers). Furthermore, queries spanning multiple data sources often are not supported, as data can only be queried when it is available within a common (or in case of the semantic web: similar) data store. For example, comparing organisms by linking BLAST results with gene ontology (GO) annotation data requires manual linking of the two data sources. Finally, once the data has been retrieved and processed, further use of the query functionality offered by the data sources is only possible by importing the data back into a data store. E.g., to perform queries on the results of a differential expression analysis requires one to put the data back into a database/triple store. Due to this overhead, most users will elect to write custom data-handling code instead. Especially within a bioinformatics research context, the mentioned limitations are encountered often.

Why do so many trivial data-handling tasks still require custom scripting solutions, while high-level data-handling query languages such as SPARQL, SQL or XPath are available? Fundamentally, the underlying cause for these problems is that both data integration and data analysis play a large role in bioinformatics. These two tasks have very different requirements. Data integration favors the absence of data structures, such as tables/matrices/nested arrays, as mapping these structures onto each other can be a difficult process. Data analysis on the other hand requires such data structures to allow for easy reasoning and aggregation across related data elements (Figure 1ab). Current query languages however do not support this complete range of data structuring, but only a limited subset. For example, RDF/SPARQL focuses on data integration, reducing datasets to collections of single facts; similarly, SQL focuses on relational tables; and XPath queries are used to query hierarchical descriptions of objects (XML). None of these query languages handle analysis-focused data structures (e.g. matrices) well. Support for data-handling operations within and between algorithms is therefore more or less absent, while this is exactly the area where it is most often needed. Therefore, most of the complex data-handling operations are still performed by the user, often by implementing them in custom written scripts.

To solve this problem, we propose a query language that can operate on data, irrespective of whether it is stored in simple or more complicated data structures. That is, we solve data-handling issues at the language level ('top-down'). Note that the bottom-up and top-down approaches are complementary: top-down needs bottom-up, as it enables easier access to (integrated) data sources and standardized identifiers, while bottom-up needs top-down as there is no universal best (agreed on) data structure, for which reason there will always be a need to 'navigate' between data structures.

Our goal has been to combine both the flexibility of low-level languages as well as the advantages of general high-level query operations. The proposed query language (which has been implemented in a prototype system called *Ibidas*) therefore uses the syntax of the Python scripting language, allowing one to mix normal scripting code with high level data operations. Besides stand-alone use, *Ibidas* also functions as middleware, allowing other analysis applications to use



**Fig. 1.** Effects of data structure and language choice. Illustrated using a microarray dataset, which contains a matrix of measurement values, a vector of gene names (related to rows of the value matrix), and a vector of tissue names (related to columns of the value matrix). a) Calculating the median distribution of gene expression values (used for microarray normalization) with Octave commands. b) Performing the same operation on data structured as a table, using SQL. Note that the 'flat' table format makes such analysis operations conceptually harder to express. c) Filtering the dataset on empty gene names. Using low level code (scripting), one has to make sure during programming that data relations remain consistent. High level data operations (queries) do this automatically.

it through webservices. This way, data-handling functions (e.g. parsers, data operations) can be shared between platforms.

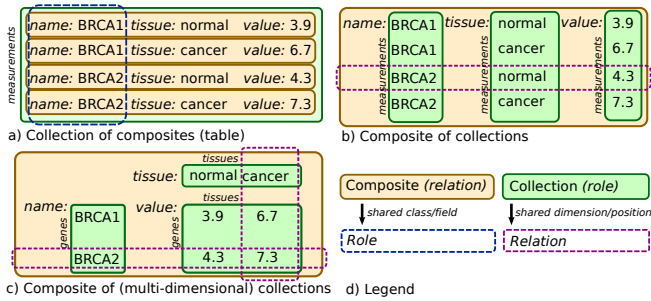
In the next section we will give an overview of the main ideas underlying this system. In section 3 we describe some aspects of the current (prototype) system in detail, followed by an extensive description of related work and a discussion in section 4 and 5.

## 2 Approach and Results

### 2.1 Annotating Data: Roles and Relations

In order to construct a query language which can operate on more complex data structures, we first focus on the question: which data representations can be queried?

Datasets consist of data elements, such as text or integers. However, without knowing the role of these data elements or the relations between the data elements, a dataset remains meaningless. For example, to assign meaning to a floating point value, both its role (e.g. it being a microarray expression measurement) as well as its relations (e.g. to a data element with the role 'gene name',



**Fig. 2.** Illustration of how role and relations are represented by composites and collections. a) Table: a collection (list) of composites (records). Composites contain mutually related data elements. Shared fields indicate a common role, e.g. 'name'. b) Inverted table: a composite of collections. Collections contain data elements with a common role. Similar positions w.r.t. to the shared dimension ('measurements') indicate mutual relation. c) Multi-dimensional collection ('values'), annotated with a 'genes' and 'tissues' dimension. Both roles and relations are still fully described.

and to a data element with the role 'tissue') have to be known. In this way, relations/roles transform data into information (see also [6]). For comparison, in the semantic web context, the equivalents of roles are properties/predicates, while relations are represented through (common) object identifiers.

The way in which such meta-data is handled forms, in our view, the main distinction between 'low-level' custom scripted data operations and 'high-level' query operations. Custom scripts require one to manually maintain relation/role consistency, while high-level data operations use relation/role information to maintain this consistency automatically (Figure 1c). One could say that scripts operate at the data level, while query operations operate at the information level.

This does require though a data representation describing both its own roles and relations. One way to annotate data in this way is the use of data structures, usually a combination of collections<sup>1</sup> and composites<sup>2</sup>. Composites group mutually *related* data elements, while collections group data elements with common *roles*.

For a queryable data structure, both roles and relations need to be represented. For composites (representing relations), one could consider that the field names indicate these roles. This is used in relational databases, where similar composites are stored in a common collection. The common collection signifies that these composites have the same role (in object-oriented terms: *class*), and by extension we can assume that similarly named fields within these composites also represent common roles. This structure is better known as a table (Figure 2a). A shared class and fields thus add the role aspect to the composite type.

<sup>1</sup> Collections: vectors, arrays, lists, e.g. [1,2,3].

<sup>2</sup> Composites: records, objects, tuples, e.g. (name='BRCA1', value=0.5).



We propose to do the same for collection types, which lack relations. Where composites can have similar field names, collections can have similar positions. Where composites can have the same class, collections can have the same ‘*dimension*’ (i.e. shared axis). Based on this parallel, we consider data elements stored in collections with a similar dimension to be related if they are stored at the same position. In this way, a shared dimension and positions add the relation aspect to the collection type.

In its most simple form, this leads to an ‘inverted table’ (Figure 2b): each collection in the composite shares the same dimension, and elements with the same positions in the different collections are related. However, one can also represent more complex cases with multiple (nested) dimensions (Figure 2c).

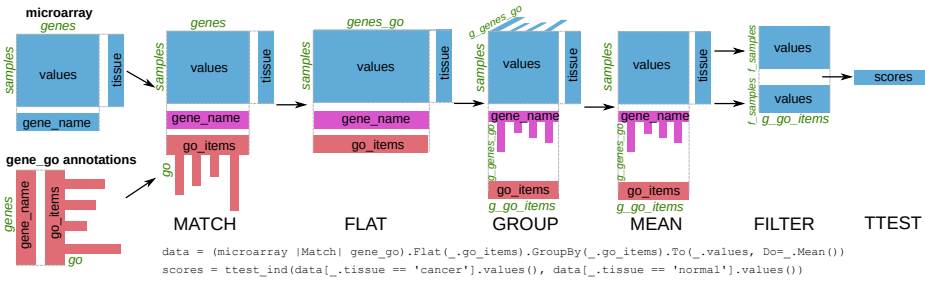
Through the introduction of shared dimensions, any (nested) data structure consisting of composites and collections can be fully annotated with relations and roles (assuming that it has only data elements with similar roles in a collection and only related data elements in a composite). With this development, a query language can be constructed which is able to query any data structure consisting of composites and collections.

## 2.2 Query Language

To construct the query language, well known data operations (e.g. Filter, Match, Group) are redefined together for use on more complex data structures. As only a single concept (‘dimensions’) was added to the data representation, we find that this can be accomplished in a relatively straightforward way. As a general rule, different collections within a dataset that have the same dimension are kept consistent with each other. If a collection is reordered or filtered, this operation is also performed on the other collections in the dataset that have the same dimension. Next to this general rule, only two extra mechanisms are required, which keeps the language simple. The first describes how an operation can target specific dimensions, and the second how operands with non-matching dimensions are handled. These are described in more detail in section 3. Here, we first illustrate the language using a few examples.

### **Example 1: Differential Expression Using Gene Ontology Categories.**

Suppose that for a certain microarray dataset, one needs to calculate differential expression not per gene, but directly per gene ontology (GO) category. The microarray data is (after normalization) available in the usual matrix format. Also, a dataset is imported containing for each gene a list of GO annotations. In Figure 3, we show how these two datasets are matched together based on common gene names, then regrouped based on GO categories, after which the expression values are averaged for each sample/GO category combination. Dimensional relations between the different vectors, matrices and nested data are automatically taken into account by the used operations, and thus do not have to be specified by the user. Due to this, all these operations can be expressed



**Fig. 3.** Calculating differential expression per GO category. Compare the steps one would have to take to implement this using a scripting language, with the shown implementation using the Ibis query language. Rectangular shapes indicate vectors, square shapes matrices, and rectangles in rectangles nested vectors. Dimension names are in *italic* and colored green. Thus, expression matrix 'values' and vector 'tissue' share the *samples* dimension. **Match** links the microarray and gene\_go dataset on common gene names. **Flat** de-nests the 'go\_items' nested vector, thereby expanding and renaming the *genes* dimension. The `_` is called the context operator, and addresses the enclosing scope (here: the result of the **Match** operation). **GroupBy** 'go\_items' groups also 'values' and 'gene\_name', due to their shared dimension. **Mean** averages 'values' along the '*g\_genes\_go*' dimension, and **Filter** is used to split the resulting matrix in a 'normal' and 'cancer' section, which are compared using t-tests. The external `ttest_ind` function (from SciPy) cannot be executed directly on Ibis query objects: adding the `()`-operator executes the query into Python data objects.

using a single line of code (shown in Figure 3), this in contrast to what would be needed if this task would have been performed manually. The output can be directly used to calculate t-tests.

**Example 2: Relating Diseases to Genes.** Given an analysis which has resulted in a number of possible cancer genes, one might want to validate the results by determining if the genes are already known as cancer genes. This requires gene-disease associations, which can be obtained from OMIM[7].

Ibis contains macros which automatically download and import such data sources. In Listing 1 it is shown how, with a few high-level operations, the data is filtered using the hypothetical cancer genes. After this, the number of associations per disease are counted and the final results are sorted based on this count. This can be done interactively, allowing the user to inspect the output at every step. Note that the code shown in Listing 1 has regular Python syntax, and can therefore be intermixed with normal Python code.

Data sources are not limited to files. The same code could have been used if OMIM was available as a webservice or database. The system would have automatically performed the webservice-calls and/or SQL queries. Furthermore, as shown in Listing 1, operations in Ibis can work with nested data structures. In most situations, this works transparently. For example, if one wants to perform

---

```

1 cancer_genes = Rep(['BRCA1', 'BRCA2'])
2 #Rep packages any Python structure into a Ibidas data object.
3 omim = Get.human.omim_genemap()
4 #predefined macro to download/import the omim genemap data.
5 #output (incomplete):
6 #gene_alias      disease
7 #[LOH18CR1, OSTS] [Osteosarcoma]
8 #[BRCA1, PSCP]   [Breast-ovarian cancer]
9 #[BRCA2, FANCD1] [Breast-ovarian cancer, Prostate cancer, Pancreatic cancer]
10
11 r = omim[(cancer_genes |Contains| omim_gene_alias).Any()]
12 #for each gene alias, determine if it is contained in cancer_genes.
13 #select genes where this is true for at least one of the gene aliases.
14 r = r.Flat(_.disease).GroupBy(_.disease)
15 #flatten the per-gene disease lists, group genes on disease.
16 #(uses context operator .. to address enclosing scope)
17 r = r.Get(_.disease, ..gene_alias.Array().Count()/".ngene").Sort(_.ngene)
18 #get the disease, count disease genes (call it "ngene"), sort on ngene
19 #Array() "packs" an array, so that we count the arrays, and not the individual aliases
20 #disease      ngene
21 #"Pancreatic cancer",      1
22 #"Prostate cancer",      1
23 #"Breast-ovarian cancer",  2

```

---

**Listing 1.** Determine which (and how often) diseases in OMIM are associated with a certain list of genes

the same task for multiple lists of genes, one could have simply replaced the cancer genes data representation with a nested array, e.g.

```
cancer_genes = Rep(['BRCA1', 'BRCA2'], ['RAD54L', 'AKT1', 'ESR1'])
```

Without changing any other line, the script still works, and results in a nested output dataset.

### 2.3 Optimization and Scalability

Performance and interactive usage are in some sense opposite goals, as interactivity suggests immediate execution of operations, while performance demands that one optimizes and executes multiple operations simultaneously.

To combine both, a lazy execution model is used: only when the user requests the contents of the data representation object, are the pending operations optimized and executed. In principle, this would allow for a fully declarative language (e.g. like SQL). However, query optimization is an open problem due to the wide range of data sources that we handle, often without any data statistics. Therefore, we chose to use a procedural approach, in which the programmer solves the problem step by step, thereby simultaneously specifying a suggested execution order. The optimizer only performs those optimizations afterwards for which it is reasonably sure that they will improve performance (e.g. not flattening/grouping the 'genes' vector in Figure 3 as it is not used in the final t-test). Although there is room for a large number of further optimizations, the current version already runs fast enough for interactive use. Even though it is written in an interpreted language, for a large expression dataset and the GO biological process annotations (54,612 probes, 180 microarrays, 442,842 gene annotations), the first line of example 1 only takes 13 seconds to execute in Ibidas. In contrast, the same query required 21 minutes when performed by a MySQL database, 128 seconds when using a PostgreSQL database, and 37 seconds when using the high performance,

column-store based, Monet Database [8] (with for the databases an optimized table design based on integer identifiers, all possible indices, and all data loaded in memory). This indicates the importance of multi-dimensional representations for efficient data analysis, as in general the efficiency of Ibidas' implementation is decidedly less efficient than the database-implementations, which use compiled languages and have been optimized for many years.

Ibidas can make use of the database query engines for data which is located in a database. This can save memory, bandwidth and time, as one can move operations to the data, instead of moving data to the operations. The lazy execution scheme makes it possible to translate (part of) a query into a data source specific query language such as SQL. This is done transparently, which has the advantage that the user does not have to learn the different query methods for the various data sources. Determining which operations a data source supports, and translating them into a query/program, is implemented using so-called wrappers [9]. Several wrappers are already available, such as the standard in-memory execution wrapper, wrappers for several commonly used biological file formats and a SQL wrapper. This design also allows for easy addition of streaming-based (memory-efficient) and parallel (time-efficient) processing in the future.

---

```

1 s = Connect('postgresql://localhost/string')
#read String database
3 i = s.network.protein_protein_links [Match| s.items.species
#couple interaction and species table on common field (species_id)
5 iy = i[_official_name == 'Saccharomyces cerevisiae']
#use only interactions occurring in yeast
7
iy = iy [Match('protein_id_a', 'protein_id')| s.items.proteins//"left"
9 iy = iy [Match('protein_id_b', 'protein_id')| s.items.proteins//"right"
#couple iy and the proteins table to get protein names for interactions.
11 #as the proteins table is used two times, assign aliases ('left', 'right')

13 imatrix_yeast = iy.GroupBy(_.left.preferred_name,
_ .right.preferred_name).combined_score.Mean()
15 #map yeast interaction scores to a 2-dimensional matrix, with for each
#protein pair an entry containing the mean score of the found interactions.
17 #(pairs without measured interactions are assigned a missing value symbol)

```

---

**Listing 2.** Obtain score matrix for yeast interactions from the String database

---

```

1 --note: aliases have been assigned readable names
SELECT prota.preferred_name, protb.preferred_name, plink.combined_score
3 FROM (((items.species AS species
INNER JOIN network.protein_protein_links AS plink ON species.species_id = plink.species_id)
5 INNER JOIN items.proteins AS prota ON plink.protein_id_a = prota.protein_id)
INNER JOIN items.proteins AS protb ON plink.protein_id_b = protb.protein_id)
7 WHERE official_name = 'Saccharomyces cerevisiae'

```

---

**Listing 3.** Automatically generated SQL by SQL wrapper for listing 2 (lines 4-12)

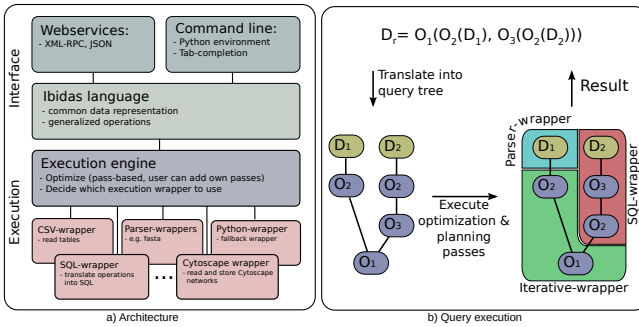
**Example 3: Loading Interaction Scores from the String Database.** As an example of how a query can be processed by multiple wrappers, protein interactions between proteins are loaded from a String database instance [10] and put into a (weighted adjacency) matrix format (Listing 2). The SQL wrapper translates the first part of the script as an SQL query (Listing 3). The second part

performs a multi-dimensional grouping (i.e. a group operation across two separate vectors, thereby creating a matrix). This cannot be translated directly into SQL, and is therefore performed internally by the default in-memory wrapper.

## 3 Methods

### 3.1 Architecture

Ibidas has a layered architecture (Figure 4a). One can interact both through a command line interface (the IPython shell [11]) as well as a webservice-interface (XML-RPC). Both interfaces use the same language layer. This language layer implements the data representation and operations, as well as the consequences of operations on the data structure / meta-data. Query execution is planned in the next layer. Here queries (combinations of operations) are converted to query trees (Figure 4b), rewritten and then executed. The rewriting is done in several passes, which are handled by a pass manager. Query execution makes use of wrappers, which are in the last layer. Here we find the actual implementation of the operations. Each type of data source has a wrapper, which describes its data structure in terms of the Ibidas data model. Furthermore, it tells the query execution layer which operations can be handled by the wrapper (either by sending it to the data source, or by implementing the functionality in the wrapper itself). Operations which are not supported by a data source wrapper are executed using the Python in-memory wrapper.



**Fig. 4.** a) An overview of the Ibidas architecture. b) A command-line query is translated into a query graph, which is optimized. Operations are assigned to execution wrappers. Subsequently, the query is executed and the result returned.

### 3.2 Data Representation

To access data, representer objects are used. These represent data source(s) and the operations that have been applied to them. Each representer consists of one or more 'slices' (which are a generalization of the column in a table). Each slice

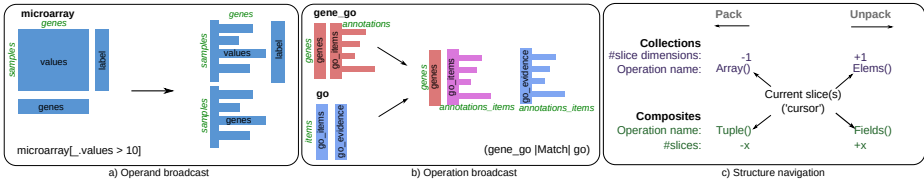
has a name, a data type, and a list of dimensions. The name represents the role of the slice, and is used to address the slice. The data type describes the structure of the individual data elements, distinguishing if it is e.g. an integer or a DNA sequence. The dimensions describe the data structure of the slice: a scalar slice has no dimensions, while a matrix has two dimensions. In general, dimensions have a fixed size, e.g. each row in a matrix has size  $n$ . However, to allow for variable sized nested data (e.g. the 'go\_items' nested vector in Figure 3), the concept of 'dependent dimensions' is used. These are dimensions whose size is variable along their parent dimensions. Using this approach, both nested and multi-dimensional structures can be handled in exactly the same way. Dimension can be shared between different slices, thereby describing the relations between the data elements in these slices. Note that dimension identities are changed by operations that change the order of elements (Sort) or shape of a dimension (Filter).

### 3.3 Data Operations

The core operations for data exploration, such as filter, groupings, joins, sorts, aggregates, and arithmetic operations are defined for 0- or 1-dimensional data. A common foundation for the execution of such operations on multi-dimensional data can be obtained through just two concepts: broadcasting and packing/unpacking.

**Broadcasting.** To be able to execute 0- and 1-dimensional operations on flexible structured data, dimensions of the operands are mapped onto each other, based on their identity. This is done for all dimensions (0-dimensional operations, e.g. addition) or all but the most nested dimension (1-dimensional operations, e.g. matching). Dimensions that cannot be mapped are 'broadcasted' to the other operands (Figure 5a). Broadcasting introduces new dimensions in an operand, by repeating data (i.e. a cartesian product): scalars become vectors, vectors become matrices, and so on. This is a well-known concept, e.g. [12]. Ibidas differs from other implementations of broadcasting in that it has the concept of dimension identity, allows for broadcasting within operations such as filtering, joining, and is able to perform broadcasting on nested arrays. Given operands that are equalized in their dimensions, the operation itself is 'broadcasted', i.e. it is applied in turn to each common operand element (0-dimensional operations) or each common vector (1-dimensional operations) (Figure 5b). Creating new operations is simplified by sharing this implementation of dimension mapping and broadcasting across operations. This way, the implementation of a new operation can remain oblivious to the full complexity of supporting the nested, multi-dimensional data model.

**Packing/Unpacking.** To prescribe at which data structure nesting level an operation has to be performed, we make use of pack/unpack operations (Figure 5c). They change at what 'structure level' subsequent operations are performed.



**Fig. 5.** a) Obtaining the genes with expression value  $> 10$  (shapes explained in Figure 3). Applying the 2-dimensional filter constraint (operand) to the 'genes' slice requires broadcasting of the *samples* dimension. Note that in this way the relation between values and gene names is maintained. b) Match (a join operation) is applied on the common 'go\_items' slices, along the *annotations* dimension. It is thus executed for each gene (i.e. it is broadcasted along the *genes* dimension). c) Navigation in nested data using pack and unpack operations. **Array** packs a dimension into a collection, while **Elms** unpacks it. Similarly, **Tuple** converts multiple slices into a single slice, by combining their related data elements into tuples (a composite type), while **Fields** unpacks such tuples.

---

```

1 #(yeast dataset from Listing 3 is used)
2 yeast.GroupBy(_.left.preferred_name)
3   .Get(_.left.preferred_name / 'protein_name',
4     _ .combined_score.Count() / 'degree',
5     _ .combined_score.Sort(descend=True)[:5].Mean() / 'top5_score'
6     ).Sort(_.top5_score)

```

---

**Listing 4.** Calculate degree and mean score of the top 5 interactions per protein, sort proteins on latter score. Note that the sort is performed for each protein, as well as over all proteins.

An example of such an operation is the **Array** function, used in Listing 1. It packs the gene aliases arrays, letting the **Count** operation count the arrays of gene aliases. The opposite operation **Elms** unpacks the arrays again. The 'packing state' is described by the slice properties. To accomplish this, we use the duality relation: **slice type**  $\leftrightarrow$  (**slice name**, **slice dimensions**). For example, unpacking a slice with a collection type converts the collection type into a slice dimension, and sets the slice type to the data type of the collection elements. Unpacking a composite data type returns for each of its fields a new slice with as name the corresponding field name and as data type the field data type. Packing performs the reverse operations, converting slice names or dimensions into data types. This is illustrated in Figure 5c. Enabling the use of operations at different structure levels allows for simpler queries. An example is given in listing 4, where we use **Sort** at different levels. Expressing such a query in a non-nested data model is much more complicated, requiring e.g. (correlated) subqueries in SQL.

## 4 Related Work

Data-handling is an actively studied topic, especially within bioinformatics. We compare Ibidas to several other approaches based on its data model, its query language, and in its role as data-handling tool/mediator.

## 4.1 Data Models

Multi-dimensional data structures [13] are popular in the context of Online Analytical Processing (OLAP), as these structures simplify data analysis. The OLAP data cube indexes tuples by measurement attributes, which form the various dimensions. The use of dimensions differs however fundamentally from Ibdidas: a data cube without measurement dimensions (i.e. a default table) is essentially 0-dimensional in the OLAP model (the relational model considers tuples to be stored in an unordered, i.e. dimensionless, set) whereas in Ibdidas, tables are 1-dimensional. More closely related are the netCDF [14] and HDF5 [15] file formats, which are widely used in e.g. the geosciences and by applications such as Matlab, to store multi-dimensional data. Similar to Ibdidas, they have the concept of dimensions (although this has not been generalized to allow for nested dimensions). Nested data models have also been studied extensively, e.g. [16–18]. Particularly interesting is the XML / XPath [19] approach here. The Ibdidas pack/unpack operations are related to the axes navigation steps that can be performed in XPath. Both nested data models and multidimensional data structures have their strengths. The main contribution of Ibdidas is that it combines table, multi-dimensional and nested approaches through the concept of dimensions.

## 4.2 Query Systems

The goal of query systems is to make data accessible through a language or other interface based on high-level operations. Standard query languages such as SQL, SPARQL and XQuery can only access one (type of) data source. In response to this, federated database tools (mediators) have been developed, which can also access other types of data sources through the use of wrappers. Examples of the latter approach are IBM Discoverylink [20], Kleisli [21], TAMBIS [22], BioMART [5], BACHIIS [23] and Biomediator [24]. In the latter four systems, individual sources are mapped against a mediated schema. Queries posed against the mediated schema can be translated to queries on the data sources. This offers an easy way to pose (declarative) queries, however it also makes adding new data sources rather complex. All of the mentioned systems work with either a language derived from SQL, and/or an API/Graphical Query Interface, which have a limited ability to handle multi-dimensional data structures. Other well-known systems, which are not directly bioinformatics related but somewhat related to Ibdidas, are Pig Latin [25] and LINQ [26]. Pig Latin is a procedural SQL derivative, enabling one to map tasks to the map-reduce framework (used for data analysis on large computer clusters). LINQ is a SQL-like query language which is embedded in the .NET framework, allowing easy access from program code. Both languages cannot be used interactively however, making them less suitable for interactive exploration of data. Prolog is a logic programming language, which can also be used as an interactive and declarative query system, offering more flexibility than standard query languages. It differs from Ibdidas in that its focus is not on enabling statistical and machine-learning-based analysis, but rather on logic-based inference. An interesting development is the proposed language



SciQL [27], whose motivations are similar to those mentioned in this work. This proposal enhances SQL for science tasks by adding support for array structures, allowing it to handle multi-dimensional arrays. Its approach differs from Ibidas, in that it adds support for data analysis tasks to the database, whereas Ibidas focuses on adding query operations to the data analysis environment.

### 4.3 Mediators and Workflow Tools

Gaggle [28] is a mediating tool, functioning as special purpose clipboard for datasets. It focuses on connecting various bioinformatics software tools and websites, by allowing them to exchange data. It is mostly oriented toward data moving, and less to performing data operations. Workflow tools are another class of data-handling tools, which depict graphically, and on a high-level, the steps that are taken in the analysis of some data. Well known examples are Taverna [29] and Galaxy [30]. Galaxy has predefined operations which one can apply to uploaded datasets in a website environment. The advantage of this approach is that it is relatively easy to use. Taverna on the other hand is a stand-alone application, allowing one to extend it with custom-scripted nodes. One can use these nodes to create a workflow graph. These nodes do not form a high-level query language though; instead they are more similar to script functions. The user remains responsible for maintaining the consistency of data relations and roles.

## 5 Discussion

To a large extent, research in bioinformatics is focused on finding new ways to combine data, by integrating and analyzing it. In this process, data management plays a central role. We have argued that current data-handling solutions mainly focus on data integration, while not adequately supporting data analysis. Two of the key problems are: a) query operations need to be supported not just on data sources, but throughout the whole bioinformatics workflow, and b) query operations should work across a range of data structures, as the best data structure is a task-specific choice, not a data-specific one. The solution proposed in this work is based on the concept of 'shared dimensions'. It is surprising to see that just adding this single concept allows for such a rich extension of the query language, enabling generic implementations of data-handling subtasks (such as filtering, grouping or matching) that support a wide range of data structures. This natural data representation also enables high query performance, by keeping track of data relations. Due to this, an implementation which in itself is not particularly fast (e.g. written in an interpreted language), can outperform even the fastest database engines on a common task such as microarray data-handling. We believe the dimension concept allows for more query language improvements than discussed in this work. For example, improved support for handling sparse data in multi-dimensional matrices would allow for the easy inclusion of all kinds of graph operations that are based on adjacency matrices. On a more long-term basis, we think the language should also incorporate various analysis algorithms

(e.g. statistical and machine learning tools) as first-class citizens, enabling their use in a standardized way. This would then truly bring together the data integration and data analysis fields in a common high-level language. Describing data relations through dimensions has opened the road towards this goal. Due to the embedding of the data-handling language within a common scripting language (Python), the user however does not have to wait for such functionality, but can easily make use of e.g. existing machine learning or graph libraries. Also, already in its current form, the ideas presented in this work could form a useful foundation on which workflow tools could be based. While we have limited our discussion to bioinformatics-based applications, as Ibdidas was designed in response to problems encountered in this field, the general approach used here may of course be equally useful for other fields in which data-intensive analytical tasks play a role.

## 6 Availability

Ibdidas has been written in the Python language. Documentation, source code and installation packages are available from the PyPI website: <https://pypi.python.org/pypi/Ibdidas>.

## References

- Galperin, M., Fernández-Suárez, X.: The 2012 nucleic acids research database issue and the online molecular biology database collection. *Nucleic Acids Research* 40(D1), D1–D8 (2012)
- Goble, C., Stevens, R.: State of the nation in data integration for bioinformatics. *Journal of Biomedical Informatics* 41(5), 687–693 (2008)
- Belleau, F., Nolin, M., Tourigny, N., Rigault, P., Morissette, J.: Bio2RDF: Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics* 41(5), 706–716 (2008)
- Goble, C., Belhajjame, K., Tanoh, F., Bhagat, J., Wolstencroft, K., Stevens, R., Nzuobontane, E., McWilliam, H., Laurent, T., Lopez, R.: BioCatalogue: a curated web service registry for the life science community. In: *Microsoft eScience Workshop 2008*, Indianapolis, IN, USA (2009)
- Smedley, D., Haider, S., Ballester, B., Holland, R., London, D., Thorisson, G., Kasprzyk, A.: BioMart – biological queries made easy. *BMC Genomics* 10(1), 22 (2009)
- Bellinger, G., Castro, D., Mills, A.: *Data, information, knowledge, and wisdom* (2004)
- McKusick, V.: Mendelian Inheritance in Man and its online version, OMIM. *American Journal of Human Genetics* 80(4), 588 (2007)
- Zukowski, M., Boncz, P., Nes, N., Héman, S.: Monetdb/x100—a dbms in the cpu cache. *IEEE Data Eng. Bull.* 28(2), 17–22 (2005)
- Roth, M., Arya, M., Haas, L., Carey, M., Cody, W., Fagin, R., Schwarz, P., Thomas, J., Wimmers, E.: The garlic project. *ACM SIGMOD Record* 25(2), 557 (1996)

10. Jensen, L., Kuhn, M., Stark, M., Chaffron, S., Creevey, C., Muller, J., Doerks, T., Julien, P., Roth, A., Simonovic, M., et al.: STRING 8—a global view on proteins and their functional interactions in 630 organisms. *Nucleic Acids Research* 37(Database issue), D412 (2009)
11. Perez, F., Granger, B.: IPython: a system for interactive scientific computing. *Computing in Science & Engineering*, 21–29 (2007)
12. Oliphant, T.: *Guide to NumPy* (2006)
13. Gysens, M., Lakshmanan, L.: A foundation for multi-dimensional databases. In: *Proceedings of the International Conference on Very Large Data Bases*, Citeseer, pp. 106–115 (1997)
14. Rew, R., Davis, G.: Netcdf: an interface for scientific data access. *IEEE Computer Graphics and Applications* 10(4), 76–82 (1990)
15. HDF Group and others: Hdf5: Hierarchical data format, <http://www.hdfgroup.org/hdf5>
16. Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., Yergeau, F.: Extensible markup language (XML) 1.0. W3C recommendation 6 (2000)
17. Colby, L.: A recursive algebra for nested relations. *Information Systems* 15(5), 567–582 (1990)
18. Kim, W.: *Introduction to object-oriented databases* (1990)
19. Clark, J., DeRose, S.: XML path language (XPath) 1.0. W3C recommendation. World Wide Web Consortium (1999), <http://www.w3.org/TR/xpath>
20. Haas, L., Schwarz, P., Kodali, P., Kotlar, E., Rice, J., Swope, W.: Discoverylink: A system for integrating life sciences data. *IBM Systems Journal* 40(2) 2001 (2001)
21. Wong, L.: Kleisli, a functional query system. *Journal of Functional Programming* 10(01), 19–56 (2000)
22. Baker, P., Brass, A., Bechhofer, S., Goble, C., Paton, N., Stevens, R.: TAMBIS-Transparent Access to Multiple Biological Information Sources. In: *Proc. Int. Conf. on Intelligent Systems for Molecular Biology*, pp. 25–34 (1998)
23. Miled, Z., Li, N., Baumgartner, M., Liu, Y.: A decentralized approach to the integration of life science web databases. *Bioinformatics Tools and Applications* 27, 3–14 (2003)
24. 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: *Workshop on Information Integration on the Web (IIWeb 2004)*, Toronto, CA (2004)
25. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig Latin: A not-so-foreign language for data processing. In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pp. 1099–1110. ACM (2008)
26. Box, D., Hejlsberg, A.: *The LINQ Project: .NET Language Integrated Query*. Microsoft Corporation (2005)
27. Kersten, M., Zhang, Y., Ivanova, M., Nes, N.: Sciql, a query language for science applications. In: *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*, pp. 1–12. ACM (2011)
28. Shannon, P., Reiss, D., Bonneau, R., Baliga, N.: The Gaggle: an open-source software system for integrating bioinformatics software and data sources. *BMC Bioinformatics* 7(1), 176 (2006)
29. 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(Web Server issue), W729 (2006)
30. Giardine, B., Riemer, C., Hardison, R., Burhans, R., Elnitski, L., Shah, P., Zhang, Y., Blankenberg, D., Albert, I., Taylor, J., et al.: Galaxy: a platform for interactive large-scale genome analysis. *Genome Research* 15(10), 1451 (2005)

# From Questions to Effective Answers: On the Utility of Knowledge-Driven Querying Systems for Life Sciences Data

Amir H. Asiaee<sup>1</sup>, Prashant Doshi<sup>1</sup>, Todd Minning<sup>2</sup>, Satya Sahoo<sup>3</sup>,  
Priti Parikh<sup>3</sup>, Amit Sheth<sup>3</sup>, and Rick L. Tarleton<sup>2</sup>

<sup>1</sup> THINC Lab, Dept. of Computer Science, University of Georgia, Athens, GA

<sup>2</sup> Tarleton Research Group, Dept. of Cellular Biology, University of Georgia, Athens, GA

<sup>3</sup> Kno.e.sis Center, Dept. of Computer Science, Wright State University, Dayton, OH  
{aha, tminning, tarleton}@uga.edu, pdoshi@cs.uga.edu,  
{satya, priti, amit}@knoesis.org

**Abstract.** We compare two distinct approaches for querying data in the context of the life sciences. The first approach utilizes conventional databases to store the data and provides intuitive form-based interfaces to facilitate querying of the data, commonly used by the life science researchers that we study. The second approach utilizes a large OWL ontology and the same datasets associated as RDF instances of the ontology. Both approaches are being used in parallel by a team of cell biologists in their daily research activities, with the objective of gradually replacing the conventional approach with the knowledge-driven one. We describe several benefits of the knowledge-driven approach in comparison to the traditional one, and highlight a few limitations. We believe that our analysis not only explicitly highlights the benefits and limitations of semantic Web technologies in the context of life sciences but also contributes toward effective ways of translating a question in a researcher's mind into precise queries with the intent of obtaining effective answers.

## 1 Introduction

Much of the data in the life sciences continues to be stored using conventional database management systems (DBMS) and subsequently, queried using the structured query language (SQL). Intuitive interfaces such as forms often provide and support “pre-canned” queries that are most commonly used by the researchers who are chiefly interested in quick and targeted accessibility to the data. However, these interfaces tend to provide more data than needed leading to time-consuming post processing steps which are specific to the local researchers, instead of being general.

We compare and contrast two approaches for querying life sciences data. Both utilize an identical data context: *strain*, *stage transcriptome* and *proteomic* data on the parasite *Trypanosoma cruzi* (*T. cruzi*). In the first approach, *T. cruzi* data is stored in a conventional DBMS and accessed through a suite of well-designed forms representing a predefined set of queries, we refer to this approach as **Paige Tools** [1] which has been the

de-facto way for storing and accessing experimental data related to *T. cruzi* by the Center for Tropical and Emerging Diseases at the University of Georgia. The second approach, *Parasite Knowledge Repository* - PKR, uses an OWL-based ontology designed in collaboration with the life science researchers to model *T. cruzi* experimental data [2]. Querying capabilities of PKR are provided by an enhanced version of a knowledge-driven querying system, *Cuebee* [3] [4], that facilitates formulation of RDF triple-based queries, which are transformed to SPARQL-DL [5].

We believe that *Paige Tools* and PKR is representative of the traditional and more sophisticated way of querying life sciences data, respectively. These approaches provide alternative ways of transforming the precise question in a researcher's mind into a computational query in order to obtain the answer. The outcome of our analysis is a set of benefits that knowledge-driven approaches such as PKR offer over the more conventional approaches. We also highlight two limitations that this approach faces, which could impede its widespread adoption despite the substantial benefits.

## 2 Related Work

Other Semantic Web based systems exist that focus on queries to provide targeted access to data in the life sciences and other contexts. These include query tools such as Openlink iSPARQL [6] and NITELIGHT [7] both of which provide graph-based interfaces for query formulation. These systems did not provide evaluation of their approaches on real-world data. Similar to PKR, GINSENG [8] offers suggestions to users, but from a different perspective. GINSENG relies on a simple question grammar, which is extended using the ontology schema to guide users to directly formulate SPARQL queries. Bernstein et al. [8] briefly evaluated GINSENG on three aspects: usability of the system in a realistic task, ability to parse large number of real-world queries, and query performance.

Semantics-based approaches also exist that focus more on data integration in the life sciences context. GoWeb [9] is a semantic search engine for the life sciences, which combines keyword-based Web search with text-mining and ontologies to facilitate question answering. GoWeb demonstrates a recall of 55 to 79% on three benchmarks. Cheung et al. [10] introduce semantic Web query federation in the context of neuroscience which provides facilities to integrate different data sources and offers either SPARQL or SQL query. Mendes et al. [4] evaluated the *usability* of *Cuebee* on the system usability scale [11] and the query formulation effort by recording time taken and number of interactions to retrieve answers. Because PKR's front end uses an enhanced version of *Cuebee* we believe that the same evaluation holds.

All of the listed approaches are available for public use. However, there is not enough evidence of how much these systems are in use by life science researchers in daily research. This paper discusses significant enhancements to *Cuebee* [3] [4], and explicitly highlights the benefits and limitations of using PKR while being used by an interdisciplinary team of computer science and cell biology researchers. Thus, while PKR is not alone in bringing knowledge-driven approaches to the life sciences, we believe that our comparative evaluation of the systems in use is novel.

### 3 Background

In this section, we briefly describe the two approaches for querying experimental data related to *T. cruzi*. We emphasize that both **Paige Tools** and **PKR** are currently operational and are being used by researchers, with the expected longer-term objective of replacing **Paige Tools** with **PKR**.

#### 3.1 Paige Tools – Conventional DBMS-Based Approach

**Paige Tools** offers interfaces to add and edit experimental data related to *T. cruzi* housed in multiple separate local databases as well as facilities to execute queries. Typically, these interfaces manifest as forms containing widgets such as drop-down lists, check boxes and buttons that allow formulation of a Boolean query on a specific dataset and selection of attributes to display in the result. We believe that the interfaces in **Paige Tools** are typical of systems utilized by life science researchers. As expressed by the researchers that use **Paige Tools**, these tend to be simple but adequate approaches for somewhat targeted access to portions of data. The interfaces are tightly coupled to the schema design and limited to executing a specific set of queries. Thus, any change to the database schema results in refactoring of the forms.

#### 3.2 PKR – Knowledge-Driven Approach

At the front end of **PKR** we use a significantly enhanced version of *Cuebee* – an ontology-based query formulation and data retrieval system applied in the context of *T. cruzi* parasite research originally designed by Mendes et al.[3] [4].

*Cuebee* employs two query engines, which we refer to as *suggestion engine* and *answer engine*. *Suggestion engine* guides a user through the process of transforming her question into a query in a logical way. It utilizes RDFS ontology schemas to suggest concepts in a drop-down list that match the characters that the user types. Furthermore, it lists all the relevant relationships for any selected particular concept. In the process of formulating the query users may need to select some intermediate concepts in order to relate the concepts that appear in the question. Finally, queries are transformed into SPARQL queries and executed by the *answer engine*.

We introduce multiple enhancements to make *Cuebee* more user-friendly [12]. For example, the enhanced *suggestion engine* now annotates each suggested concept with information that includes a description of the ontology class and associated properties. It allows selection of multiple instances that satisfy Boolean operators. The enhanced *Cuebee* also guides users to formulate more complex SPARQL graph patterns using group by and aggregate functions, filter over instances using regular expressions. In addition, an undo feature helps users revise their queries at any point during the formulation process.

Our contributions go beyond the interface and focus on the infrastructure of *Cuebee* as well. A major improvement is the capability to support OWL ontologies because they tend to be more expressive than RDFS ontologies. For example, in the context of *T. cruzi* research, we use the OWL-based parasite experiment (PEO) and

parasite lifecycle (OPL) ontologies [2]. Subsequently, we equip the two query engines to execute SPARQL-DL [5] queries which offer more expressive power than SPARQL. OWL ontologies are deployed in an OWL-DL reasoner called *Pellet* in order to take advantage of the inferencing capabilities.

An increasing number of bioinformatics tools and biomedical data sources are available as Web services. As another contribution to *Cuebee*, we extend the results of the final queries with common bioinformatics tools such as EBI BLAST available as RESTful Web services and access into TriTrypDB [13]. Here, we detect if the results of a query contain appropriate types of protein sequences or gene IDs, and allow the user to trigger an invocation of the EBI BLAST Web service or obtain additional information from TriTrypDB.

## 4 Benefits of PKR over Paige Tools

Both Paige Tools and PKR are running concurrently on identical data and in use by a team of researchers. The identical contexts provide us a valuable opportunity to comparatively evaluate the two approaches in a principled way in this section.

### 4.1 Explicitly Structured Queries

The first benefit is with respect to the structure of the queries that may be formulated in the two approaches. In order to illustrate this, consider the following question posed by parasitologists in the context of *T. cruzi*:

*Which microarray oligonucleotide derived from homologous genes has 3 prime region primers?*

Note that homology is a relationship between two genes (these genes are derived from a common ancestor) and *3-prime-region* is a property of primers.

Conventional database design places minimal importance on named relationships (e.g., table joins) and Paige Tools as a typical example of DBMS-based systems that are in use in life science research labs, reflects this. While query pages within Paige Tools provide users the ability to show attributes of *microarray oligonucleotide*, *genes* and *primers*, discerning homology relationships between two genes is left to the ability of the user in post-processing the results. Thus, the resulting query does not adequately reflect the original question in the researcher’s mind.



**Fig. 1.** Formulated query for “Which microarray oligonucleotide derived from homologous genes has 3 prime region primers?” in PKR. Notice the relationships between the concepts.

On the other hand, PKR’s process of formulating queries allows a logical interpretation of the question. Queries formulated within PKR contain not only the concepts (e.g., *gene*) but also make the relationships explicit in the query (e.g., *is homologous to*), as we show in Fig. 1. The query formulation process in PKR leads

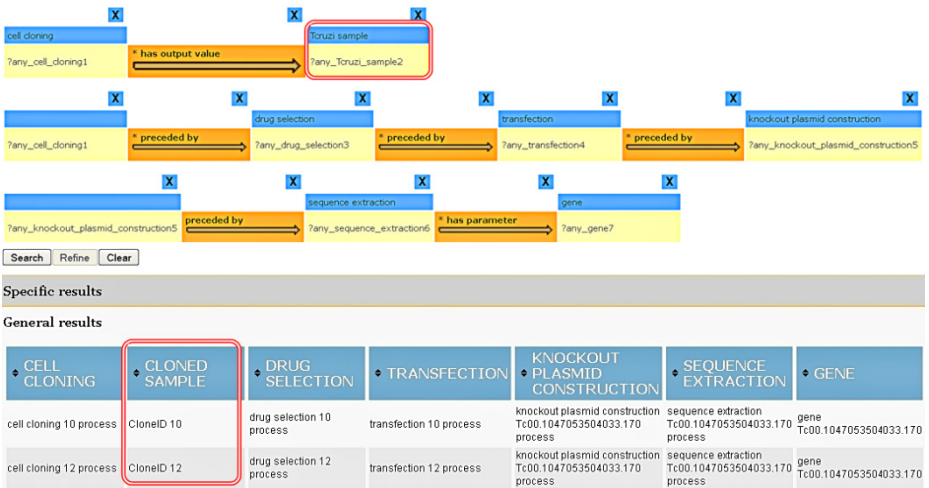
users to find linkages between concepts by suggesting relationships explicitly. Due to the expressiveness of ontology schemas, the formulated query is more readable and promotes better understanding even to users with less domain knowledge.

### 4.2 Queries at Different Levels of Abstraction

A significant benefit of PKR is its ability to query at multiple levels of abstraction. This is beneficial because researchers investigating new hypotheses often ask general questions. Consider the following question posed by our parasite researchers:

*What genes are used to create any T. cruzi sample?*

*T. cruzi sample* could be of several different types: *cloned sample*, *drug selected sample*, and *transfected sample*. There is no straightforward way to transform this general question into a query using Paige Tools. A researcher translates this question into a query for strains database that produces almost all genomic data. Then, the researcher tediously analyzes multiple attributes for each data record to ascertain the type of *T. cruzi sample*. In this approach, explicitly linking the different samples would involve redesigning the database and reduced efficiency.



**Fig. 2.** The question “What genes are used to create any *T. cruzi sample*?” is formulated in PKR and *cloned sample* which is a type of *T. cruzi sample* appears in the results

On the other hand, PKR intuitively models the relationships between the different types of samples in the ontology schema. PKR’s *answer engine* takes advantage of Pellet’s inferencing by using SPARQL-DL’s extended vocabulary and generates the corresponding query in order to access instances of the class and all its subclasses. As Fig. 2 illustrates, *cloned sample* – a subclass of *T. cruzi sample* – appears under the “General Results” tab. Therefore, answering general questions is less dependent on a user’s domain expertise in contrast to Paige Tools.



### 4.3 Uniform Query Interface

Ontology-driven approaches such as PKR allow a uniform query interface for multiple related datasets; however, Paige Tools offers several interfaces to access the different databases. Each interface is designed using drop-down lists holding different attribute names from the corresponding table schema and check boxes to give the option to the user of filtering results (see Fig. 3). Notice that the items in the drop-down lists and the check box labels differ across the two interfaces.

PKR provides a uniform query interface to the user regardless of which datasets are the target of the questions. The process of translating the question into a query does not change with different contexts. By default, formulated queries are executed over all datasets. Users may also select a suitable dataset from the drop-down list of datasets for efficiency. This is enabled by using a single, comprehensive ontology schema for the related datasets. Furthermore, approaches such as PKR are usually not tied to a specific ontology but support any ontology designed in OWL.

**Gene Annotation Queries.**

Please select a boolean condition or describe a query

AND | OR | NOT

Gene ID (or ORF id) LIKE Tc00

Other Feature Query

If searching genes for features:  Gene Name  Annotation Data

Evaluate expression as:  Exists or  Not Exists

Extend bounding box of ORF by: 0 bases

Please select a boolean condition or describe a query

AND | OR | NOT

none LIKE

Start Over!

Submit Query!

Select the number of results per page: 25

Column Name	Show
Gene Size	<input checked="" type="checkbox"/>
Gene ID (or ORF id)	<input checked="" type="checkbox"/>
Gene Name	<input checked="" type="checkbox"/>
Annotation Data	<input checked="" type="checkbox"/>
Feature Type	<input type="checkbox"/>
Feature Description	<input type="checkbox"/>

**(a)**

**Gene Cloning Database Query Page**

Please select a boolean condition or describe a query

AND | OR | NOT

Clone Name LIKE Tc00.1047

OR

AND | OR | NOT

Clone Designation LIKE Tc00.1047

AND | OR | NOT

Clone Designation LIKE

Start Over!

Submit Query!

Column Name	Show
Clone Designation	<input checked="" type="checkbox"/>
Clone Name	<input checked="" type="checkbox"/>
Gene Homo Acc	<input checked="" type="checkbox"/>
Product Size (NA)	<input checked="" type="checkbox"/>
Product Size (AA)	<input checked="" type="checkbox"/>
Product Weight	<input type="checkbox"/>
Cloning Status	<input type="checkbox"/>
Notes	<input type="checkbox"/>
GSS Accession	<input type="checkbox"/>
Working Location	<input type="checkbox"/>
Archive Location	<input type="checkbox"/>
Forward Primer	<input type="checkbox"/>
Reverse Primer	<input type="checkbox"/>
Annotator	<input type="checkbox"/>
Cloning Pool	<input type="checkbox"/>

**(b)**

**Fig. 3.** The (a) gene annotation query page and (b) cloning database query page – representing two interfaces of Paige Tools

### 4.4 Querying over Multiple Datasets

Often, researchers pose questions that span across different types of data. For example, consider the following question:

*Which genes with log-base-2-ratio greater than 1 have 3 prime region primers?*

Data related to *log-base2-ratio* is found in the *transcriptome* dataset while primers with *3-prime-regions* are found in *strain* dataset. In Paige Tools question is divided into two sequential sub-questions: (a) *Which genes have log-base-2-ratio greater than 1*; and (b) *which of these genes has 3-prime-region primers*. Answer to question (a) is found using the gene annotations query page. Then, a researcher takes the results from (a) and manually looks for the primers in the gene cloning page to find answers to (b).

On the other hand, PKR allows a formulation of the associated query without decomposing it despite the fact that two different datasets hold the answers. A user finds



**Fig. 4.** The question, “Which genes with log-base-2-ratio greater than 1 have 3 prime region primers”, formulated in PKR. The query for this question spans multiple datasets.

the appropriate concepts and relationships between *log-base-2-ratio* and *gene* (Fig. 4 area (1)), and continues to formulate the query by adding the *has-3-prime-region* relationship followed by *region* (Fig. 4 area (2)). On formulating the query, PKR allows a search over all datasets – made possible because of a comprehensive ontology for all the data. The solution to the query integrates both datasets thereby facilitating integrated analysis by the researchers with minimal post-processing effort.

## 5 Limitations of PKR

We highlight two limitations of approaches such as PKR, which may likely impact its widespread adoption. While ontologies represent a formal model of the domain knowledge, users not well acquainted with the ontology feel tied down to its structure. We minimize this by providing suggestions about next possible concepts and relationships. Nevertheless, our triple-based queries often require users to select intermediate concepts and relationships that connect the entities in the question. But users prefer more abbreviated queries in their daily usage of systems such as PKR.

The second limitation is the increased time and space complexity of knowledge-driven systems compared to highly optimized modern DBMS. While fast RDF storages such as *Virtuoso* exist, the predominant complexity is due to the ontology inferencing facilities provided by systems such as *Pellet*.

## 6 Evaluation and Discussion

While Mendes et al. [4] evaluated the usability of PKR’s interface, in this paper, we focus on the *usefulness* of knowledge-driven systems such as PKR in comparison to DBMS-based systems such as *Paige Tools*, which requires that the systems be in use. We compile our observations of both systems in use into the benefits and limitations of the two approaches, in Sections 4 and 5. In order to quantify aspects of usefulness of PKR and *Paige Tools* we calculate precision and recall on a corpus of 25 domain questions, many of which span multiple datasets. Although the domain of these questions is limited to the parasite, *T. cruzi*, such questions are commonly encountered by biologists and parasitologists investigating other organisms as well.

Two domain experts independently validated the consensual reference set for each question in this evaluation. We obtain average precisions of 83% and 56% for PKR and *Paige Tools*, respectively; average recall score for PKR is 80% and for *Paige Tools* is 77%. Our results show that both systems retrieve large fractions of the relevant data

from the collection of all data, and queries in PKR provide more accurate answers than in Paige Tools. The latter lead to much post processing, as mentioned.

Parasitologists using PKR appreciate its advantages and are getting more comfortable with the layout as it improves. But, it takes time to get researchers to change over completely. We are not yet at a point where researchers in other labs may be able to simply install PKR and query their particular sets of data. Many of the concepts used in PEO are general enough to be incorporated into ontologies for other organisms, but we anticipate that ontologies will still require tailoring to individual use cases. The scope of this paper is to provide a model for developing ontology-based systems for life science researchers, to offer proof that semantic Web technologies will ultimately be of greater use to biomedical researchers than traditional DBMS, and to demonstrate the capabilities of PKR. We believe that these are substantive steps towards developing systems that are more user friendly and efficient for biomedical researchers. As PKR continues to be utilized we expect that researchers will gain new biological insights from their analysis of the data.

## Reference

1. Paige tools, <http://paige.ctegd.uga.edu>
2. Parikh, P., Minning, T., Nguyen, V., Lalithsena, S., Asiaee, A., Sahoo, S., Doshi, P., Tarleton, R., Sheth, A.: A Semantic Problem Solving Environment for Integrative Parasite Research: Identification of Intervention Targets for *Trypanosoma cruzi*. *PLoS Neglected Tropical Diseases* 6(1), e1458 (2012)
3. Cuebee (Original Version), <http://Cuebee.sourceforge.net>
4. Mendes, P., McKnight, B., Sheth, A., Kissinger, J.: Tcrucikb: Enabling complex queries for genomic data exploration. In: *IEEE-ICSC*, pp. 432–439 (2008)
5. Sirin, E., Parsia, B.: SPARQL-DL: SPARQL Query for OWL-DL. In: *Third OWL Experiences and Directions Workshop (OWLED)* (2007)
6. Kiefer, C., Bernstein, A., Lee, H.J., Klein, M., Stocker, M.: Semantic Process Retrieval with iSPARQL. In: Franconi, E., Kifer, M., May, W. (eds.) *ESWC 2007*. LNCS, vol. 4519, pp. 609–623. Springer, Heidelberg (2007)
7. Russell, A., Smart, P., Braines, D., Shadbolt, N.: NITELIGHT: A Graphical Tool for Semantic Query Construction. In: *SWUI Hosted by CHI*, Florence (2008)
8. Bernstein, A., Kaufmann, E., Kaiser, C., Kiefer, C.: Ginseng: A Guided Input Natural Language Search Engine for Querying Ontologies. In: *Jena User Conference*, UK (2006)
9. Dietze, H., Schroeder, M.: GoWeb: a semantic search engine for the life science web. *BMC Bioinformatics* 10(suppl. 10), S:7 (2009)
10. Cheung, K., Frost, H., Marshall, M., Prud'hommeaux, E., Samwald, M., Zhao, J., Paschke, A.: A journey to Semantic Web query federation in the life sciences. *BMC Bioinformatics* 10(suppl. 10), S:10 (2009)
11. Brooke, J.: SUS: a quick and dirty usability scale. In: *Usability Evaluation in Industry*, pp.189–194 (1996)
12. Cuebee (Enhanced Version), <http://jade.cs.uga.edu:8080/Cuebee>
13. TriTrypDB, <http://tritrypdb.org>

# OmixAnalyzer – A Web-Based System for Management and Analysis of High-Throughput Omics Data Sets

Thomas Stoltmann\*, Karin Zimmermann\*, André Koschmieder\*, and Ulf Leser

Humboldt-Universität zu Berlin, Germany  
Department of Computer Science

{stoltman, zimmer, koschmie, leser}@informatik.hu-berlin.de

**Abstract.** Current projects in Systems Biology often produce a multitude of different high-throughput data sets that need to be managed, processed, and analyzed in an integrated fashion. In this paper, we present the OmixAnalyzer, a web-based tool for management and analysis of heterogeneous omics data sets. It currently supports gene microarrays, miRNAs, and exon-arrays; support for mass spectrometry-based proteomics is on the way, and further types can easily be added due to its plug-and-play architecture. Distinct from competitor systems, the OmixAnalyzer supports management, analysis, and visualization of data sets; it features a mature system of access rights, handles heterogeneous data sets including metadata, supports various import and export formats, includes pipelines for performing all steps of data analysis from normalization and quality control to differential analysis, clustering and functional enrichment, and it is capable of producing high quality figures and reports. The system builds only on open source software and is available on request as sources or as a ready-to-run software image. An instance of the tool is available for testing at [omixanalyzer.informatik.hu-berlin.de](http://omixanalyzer.informatik.hu-berlin.de).

## 1 Introduction

Current projects following a Systems Biology approach to the study of biomedical phenomena typically produce a multitude of different high-throughput data sets. For instance, to study complex phenotypes such as cancer [6] and other genetic diseases [1], researchers analyze cellular samples at various levels, such as gene expression, protein expression, epigenetic status of regulatory elements in the genome, presence of differentially spliced protein isoforms, levels of metabolites etc. Managing and analyzing such diverse and heterogeneous data sets is a significant challenge; therein, analysis cannot stop at individual data sets, but needs to intelligently combine data generated by different methods [12]. In concrete projects, such technical and scientific issues are engrained by more social issues, such as highly different levels of proficiency of project members with modern methods in data analysis, problems in terms of data sharing, and unclear separations of concern between experimentalists and bioinformaticians [3].

---

\* These authors contributed equally to this paper.

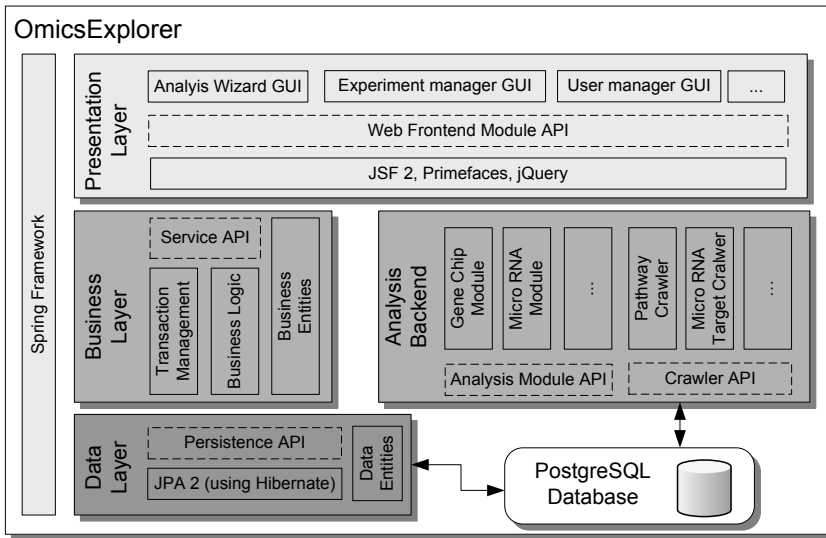
The degree to which this harms projects varies with the size: Typically, small projects (2-4 groups) often have a clear separation of duties, a modest heterogeneity in data sets and less problems with data sharing. In large, international or national-wide projects such issues are usually resolved in a strict and managed manner by enforcing agreed policies and hierarchical organization [16]. However, the majority of typical Systems Biology projects probably are just in-between, uniting 10-20 groups with mixed expertise, having different perspectives on their subject, and gathering a heterogeneous set of experimental data. Further, these groups are often very sensitive to questions of data ownership and access authorization. Especially projects of this size can greatly benefit from central solutions to data management and analysis, as they may help to reduce cost, establish unified standard operating procedures, and foster data exchange [10].

In this paper, we present OmixAnalyzer, a system for managing, analyzing, sharing and visualizing heterogeneous -omics data sets for mid-sized projects. It uses a central database to store experimental results and sample metadata, features a full blown three-tier access rights management, and offers an intuitive web interface tailored towards biologists without specialized computer training. The OmixAnalyzer uses only open source components and builds on a plug-in architecture for adding novel data types with their individual metadata, internal data structures, and workflow-based analysis pipelines. For data analysis, the system executes configurable pipelines of R scripts, which makes changes in terms of individual analysis tools or the way algorithms are combined quite easy. Analysis results are downloadable in spreadsheet formats, can be used to generate publication-quality figures, and are stored by the system for later reuse.

We believe that the feature set of OmixAnalyzer is quite unique. For instance, systems such as Intermine [14] or Galaxy [5] are focused on genome sequences and do not target dynamic data sets typical for transcriptomics or proteomics. A variety of tools exist for transcriptome data [8], but none is suitable for our setting; for instance, using Chipster requires uploading data to a project-external server [7], while Mayday [2] clearly targets only expert bioinformaticians. Systems explicitly focusing on multiple omics data are, for instance, Vanted [11], which focuses on graph-based analysis and visualization, or Babelomics [9], which targets nation-wide projects and lacks a data access model. A number of other projects, such as SysmoSeek [15] or DIPSBC [4] only target data management but not data analysis or visualization. Overall, we are not aware of any other system that specifically targets mid-size Systems Biology projects.

## 2 System Architecture

The system architecture of the OmixAnalyzer follows the Model-View-Controller pattern in most cases and is segmented into three different layers: presentation, business, and data layer. The presentation layer contains all GUI elements, i.e. the web pages the user navigates when using the system. The business layer contains the operational logic, handles the communications between the front-end and the data layer, and manages the database transactions. Also in this



**Fig. 1.** System architecture of the OmixAnalyzer and software used

layer is the analysis back-end, which contains all analysis modules available in the OmixAnalyzer as well as the data crawlers. The data layer handles the data persistence and communications with the database.

Note that the software project completely relies on open source software. Figure 1 gives an overview of the system architecture and shows its components.

The layers have been designed in a modularized manner, with each module covering a specific task. This way, new functionality can easily be added to the system by means of adding new modules. Details on how to add new modules (e.g. new data types) are covered in Section 5. In the presentation layer, a module represents web pages like user settings, experiment manager or analysis wizard.

The analysis backend consists of modules for data analysis and data crawlers. A data analysis module contains all available analysis workflows for a specific platform, with gene chips, micro RNA, and exon arrays currently available. A module exports description files for all supported analysis workflows providing information on required data and options for the workflow. For example, supported analysis workflows for the gene chip module are Clustering, Differential Analysis, Functional Analysis, Quality Control and Visualization. All currently implemented workflows use R as analysis backend, and methods for invoking R and handling jobs are available in the analysis API. However, new analysis modules are not required to use R as backend, but can also use Java, Perl, or other languages.

A data crawling module provides the system with data gathered from external resources. Crawlers can be run manually, or automatically at specific intervals.

The crawlers provide annotation data used in the analysis workflows as well as pathway and gene identification data. Currently, the following data sources are supported: BioMART, KEGG (the last publicly available version), BioGRID, Reactome, TarBase, MirTarBase, MicroCosm, Miranda, PicTar and mapping files of different manufacturers. Additional crawling modules can easily be added.

The data layer is responsible for storing the data used in OmixAnalyzer. While some data formats like raw experiments or binary normalized expression data are stored in the file system, most data is stored in the relational PostgreSQL database. This includes all experiments and their annotations, users, groups and permissions, crawled external data, and analysis jobs and results.

### 3 Supported Data Analysis

A short overview of the functionality of the OmixAnalyzer demonstrates the variety of analysis possibilities. On top of supporting both the analysis of microarray and sequencing technologies, their joint analysis is a key feature of the presented software.

The analysis possibilities provided are organized in six workflows (see Figure 2(b)) according to their main topics for better overview and user guidance: Quality control, differential analysis, clustering, visualization, functional analysis and joint analysis.

Quality control implements commonly used plots such as boxplot, array-array correlation plot or PCA which enable the user to estimate the quality of the data and detect potential outliers. Visualization contains a potpourri of widely used plots providing a powerful tool for investigative and hypothesis generating analyses on the one hand as well as the visualization of results on the other hand. Clustering analysis can be applied to samples or genes, or both. Hierarchical clustering using the complete linkage algorithm and euclidian distance provides a generic and powerful tool in class discovery. Differential analysis is a state-of-the-art implementation facilitating the detection of relevant genes based on commonly used criteria such as t-test based p-value, fold change or gene expression variance. For more than two groups, Anova is available.

A more bio-functional interpretation of the results can be obtained with Functional analysis. A pre-selected set of genes, coming from differential analysis or selected by other criteria, can be tested for significant enrichment of KEGG pathways or GO terms. This service is provided for all microarray-based data that originated from chips supported by Bioconductor. The joint analysis enables the user to compare and correlate data from different technologies as long as they contain samples which can be matched. Based on different criteria, subsets of data can be selected and combined for correlation and visualization.

Each workflow leads to a result page (see Figure 2(a)) containing both a pdf file incorporating all analysis results as well as all single result files for download.

Additionally, every workflow provides highly specialized filtering options with respect to the data used for the concrete analysis. Clustering for example can be

**Manage your analyses****Differential Analysis - View Analysis Results**

Your Job is complete. You can view and download all job result images to view them in your Browser, or use the links to download

**What to do next?**

- Run a new analysis
- Download all results as a compressed archive file

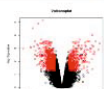
**Heatmap**



A heatmap is a color coded visualization of numerical data. Here, the column and rows are clustered together.

Download to your computer (png format)  
Download to your computer (pdf format)

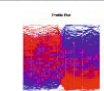
**Volcano Plot**



Visualize differential expressed genes by plotting the p-value (y-axis).

Download to your computer (png format)  
Download to your computer (pdf format)

**Profile Plot**



Profile plots are line plots that display the difference between the different samples.



Download to your computer (png format)  
Download to your computer (pdf format)

**Fig. 2(a).** Downloadable analysis results for differential analysis.

Home > Select Data and Analysis Workflow > Set Parameters > Run Analysis

Select one or more experiments to analyze.

**Your selected experiments**


Main	Name	Action
<input checked="" type="checkbox"/>	GSE3678	 

Add another experiment

**Available analysis workflows**

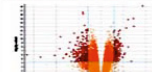
1 12

**Clustering Analysis**




Grouping similar elements helps identifying new classes of samples.

**Differential Analysis**




Identify genes with significant behaviour between sample classes.

**Visualization**




Identify significant samples.

**Functional Analysis**



Visualize data using profile-plots, heatmaps, SVD or histograms.

**Quality Control**



Visual overview of the data using QC plots like boxplot or MA-Plot.

1 12

**Fig. 2(b).** Workflows provided for the selected dataset

applied to samples, genes or both. Differential expression analysis will produce results for genes meeting certain criteria such as p-value or simply return a list and figures, and calculating the test statistic for all selected genes. A further filtering option is based on the role of genes. The user can select the latter based on their membership in a certain pathway or by user-defined lists. Microarrays can even be selected based on their target genes. Due to the almost infinite number of combinations the filtering techniques make the OmixAnalyzer a very powerful tool by providing a maximum of creative analysis freedom.

A very straightforward but flexible and effective way to implement integrated data analysis is provided by the option to store gene lists at the end of an analysis workflow and reuse the entities, i.e. genes, in another. If, for example, epigenetic as well as expression data of the same conditions were stored in the OmixAnalyzer, the set of differentially expressed genes could be tested for differential histone acetylation. Sample correspondence over experiments permits even more sophisticated options, such as miRNA to mRNA mapping by target as well as their correlation.



## 4 Web Interface

The web interface is entirely written in Java and XML and relies on JSF2, Primefaces, HTML5 and Ajax to offer a seamless user interface. Ajax is implemented through partial page rendering and partial page processing. Most of the client-side JavaScript code is provided through the frameworks used. The web interface is theme-able.

All administration of OmixAnalyzer can be done in the web interface by a user with administration privileges. The so-called administration area provides pages to manage entities within the system. The Experiment Manager Module allows to create, edit, delete and export experiments. The creation of new experiments can be done entirely using the web interface, including the upload of the experiment data. Experiments can be exported as a single zip file containing the data of the experiments in mage-tab format as well as the raw data (e. g. cel files of gene chip experiments). The user manager module contains facilities to create, edit and delete users and their roles. Additional modules are available to manage platforms, organisms, etc.

A more illustrative impression of the web interface shall be achieved by a walk-through to joint Analysis of gene chips and miRNA. The step preceding all workflows is the selection of the data set. Based on this decision, the workflows available for the selected data sets are displayed. In case of joint analysis, a miRNA and a gene chip data set need to have corresponding samples. After choosing the workflow, a subset of groups can be selected. For a more specific analysis a filter can be applied to genes as well as to miRNAs. The emerging subset of samples and targets is now correlated based on an internally provided miRNA target mapping. Once the calculations are complete, the result page offers the view and download of all generated results including images in a pdf file, by single download or as an all-including compressed archive.

## 5 Availability and Extensibility

The OmixAnalyzer is available on request. We provide the full sources required to build and run the system, as well as a ready-to-run virtual machine image. Because the OmixAnalyzer is easy to administer but complex to install, this image can be used to set-up the system in only a few steps. The system is also available online for trying out its features: <http://omixanalyzer.informatik.hu-berlin.de>

To run the OmixAnalyzer from the virtual machine image, VirtualBox (4.2 or higher) is required, which is freely available for most operating systems. The image contains all required software, and an installation guide for setting up the virtual machine is also provided.

Compiling and deploying OmixAnalyzer requires a few more steps. The system presupposes a Linux system with Oracle's Java JDK 7, PostgreSQL (9.0 or higher), R (2.15 or higher) and Bioconductor, ImageMagick and Maven 3. After modifying the central configuration file according to your needs, Maven will automatically download all required libraries and build a ready-to-run war file,

which can be deployed on a Tomcat servlet container (version 7 or higher) or a Glassfish application server (3.1 or higher). The database can easily be installed using the supplied schema files. A detailed installation guide is also available.

With the modular architecture, new components can be added to the system with relatively small effort as plug-ins. These can add new GUI components to the system, provide more analysis methods for supported platforms, or even add a completely new platform type like for example proteomics.

To add a new GUI component, only the website layout and a Java class containing the GUI related functionality need to be written. For new platform types, a small module containing platform-specific information is required, along with some minor changes in the Data Layer. To enable users to analyze data of the new platform type, new analysis modules are required as well.

Analysis modules are the most complex type of plug-ins for OmixAnalyzer. A new analysis module must contain meta information about the analysis workflow, including required input data and user options to parameterize the analysis. The actual analysis functionality can be supplied as Java, R, Perl or similar code routines.

## 6 Discussion

We present the OmixAnalyzer, a system for data management and analysis that specifically targets the needs of mid-size projects. For smaller groups of people, maintaining a central solution like the OmixAnalyzer probably brings more burden than gain; in such settings, stand-alone systems, some of which are also available commercially, are usually the better choice. On the other hand, very large, international projects typically need a more flexible system than the OmixAnalyzer, with stronger capabilities in terms of scalable and distributed data analysis, support for automatic data and metadata ingestion, and possible direct links to LIMS systems. The OmixAnalyzer was designed to target projects just in-between, which typically can afford (limited) central and professional staff for running a data management solution.

The system supports both computer-illiterate and savvy users. Biologists can take advantage of the built-in pre-processing, quality-control and data analysis options to work with their data and to generate results and figures for publications. Bioinformaticians may exploit the plug-in architecture to easily adapt the system to specific needs or to add specific extensions, like novel analysis pipelines or support for other data types. The successful use of the OmixAnalyzer manifests in at least two publications [6,13] accrued in the framework of the TRR54.

We are currently extending the system to also support proteomics data. Furthermore, we plan to provide a more comprehensive set of joint analysis methods.

**Acknowledgements.** We acknowledge funding from the Deutsche Forschungsgemeinschaft (DFG) through Transregio TRR-54. We thank Y. Mayer, L. Sousa and M. Pichotta for contributions, and M. Hummel and other Transregio members for feedback.

## References

1. Baranzini, S.E., Mudge, J., van Velkinburgh, J.C., Khankhanian, P., Khrebtukova, I., et al.: Genome, epigenome and rna sequences of monozygotic twins discordant for multiple sclerosis. *Nature* 464(7293), 1351–1356 (2010)
2. Battke, F., Symons, S., Nieselt, K.: Mayday – integrative analysis for expression data. *BMC Bioinformatics* 11, 121 (2011)
3. Birney, E., Hudson, T.J., Green, E.D., Gunter, C., Eddy, S., et al.: Prepublication data sharing. *Nature* 461(7261), 168–170 (2009)
4. Dreher, F., Kreitler, T., Hardt, C., Kamburov, A., Yildirimman, R., et al.: Dipsbc-data integration platform for systems biology collaborations. *BMC Bioinformatics* 13(1), 85 (2012)
5. Goecks, J., Nekrutenko, A., Taylor, J.: Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol.* 11(8), R86 (2010)
6. Joosten, M., Seitz, V., Zimmermann, K., Sommerfeld, A., Berg, E., et al.: Histone acetylation and dna demethylation of t-cells result in an anaplastic large cell lymphoma-like phenotype. *Haematologica* 98(2), 247–254 (2013)
7. Kallio, M.A., Tuimala, J.T., Hupponen, T., Klemela, P., Gentile, M., et al.: Chipster: user-friendly analysis software for microarray and other high-throughput data. *BMC Genomics* 12, 507 (2011)
8. Koschmieder, A., Zimmermann, K., Trissl, S., Stoltmann, T., Leser, U.: Tools for managing and analyzing microarray data. *Brief. in Bioinf.* 13, 46–60 (2012)
9. Medina, I., Carbonell, J., Pulido, L., Madeira, S., Goetz, S., et al.: Babelomics: an integrative platform for the analysis of transcriptomics, proteomics and genomic data with advanced functional profiling. *Nucl. Acids Res.* 38(suppl.) W210–W213 (2010)
10. Paton, N.W.: Managing and sharing experimental data: standards, tools and pitfalls. *Biochem. Soc. Trans.* 36(Pt. 1), 33–36 (2008)
11. Rohn, H., Junker, A., Hartmann, A., Grafahrend-Belau, E., Treutler, H., et al.: Vanted v2: a framework for systems biology applications. *BMC Syst. Biol.* 6(1), 139 (2012)
12. Schadt, E.E., Linderman, M.D., Sorenson, J., Lee, L., Nolan, G.P.: Computational solutions to large-scale data management and analysis. *Nat. Rev. Genet.* 11(9), 647–657 (2010)
13. Seitz, V., Thomas, P., Zimmermann, K., Paul, U., Ehlers, A., et al.: Classical hodgkin’s lymphoma shows epigenetic features of abortive plasma cell differentiation. *Haematologica* 96(6), 863–870 (2011)
14. Smith, R.N., Aleksic, J., Butano, D., Carr, A., Contrino, S., et al.: Intermine: a flexible data warehouse system for the integration and analysis of heterogeneous biological data 28(23), 3163–3165 (2012)
15. Wolstencroft, K., Owen, S., du Preez, F., Krebs, O., Mueller, W., et al.: The seek: a platform for sharing data and models in systems biology. *Methods in Enzymology* 500, 629 (2011)
16. Wruck, W., Peuker, M., Regenbrecht, C.R.: Data management strategies for multinational large-scale systems biology projects. *Brief. in Bioinf.* (2012)

# The RDF Pipeline Framework: Automating Distributed, Dependency-Driven Data Pipelines

David Booth

Independent Consultant

KnowMED, Inc.

david@dbooth.org

<http://dbooth.org/2013/dils/pipeline/>,

<http://rdfpipeline.org/>

**Abstract.** Semantic web technology is well suited for large-scale information integration problems such as those in healthcare involving multiple diverse data sources and sinks, each with its own data format, vocabulary and information requirements. The resulting data production processes often require a number of steps that must be repeated when source data changes -- often wastefully if only certain portions of the data changed. This paper explains how distributed healthcare data production processes can be conveniently defined in RDF as executable dependency graphs, using the RDF Pipeline Framework. Nodes in the graph can perform arbitrary processing and are cached automatically, thus avoiding unnecessary data regeneration. The framework is loosely coupled, using native protocols for efficient node-to-node communication when possible, while falling back to RESTful HTTP when necessary. It is data and programming language agnostic, using framework-supplied wrappers to allow pipeline developers to use their favorite languages and tools for node-specific processing.

**Keywords:** Data flow, data pipelines, semantic web, RDF, SPARQL.

## 1 Introduction

A major use case for semantic web technology in industry is information integration involving several diverse data sources, each having its own access protocols, data format, vocabulary and information content. Healthcare data fits this profile well. When semantic web technology is used for this purpose, source data such as patient information and lab data must be accessed, converted to RDF[1], and transformed in ways that are specific to each data source, to link the information together. Ontologies and rules are useful in performing semantic transformation of the information, and often require multiple processing steps. In addition, if the information is important – such as healthcare information – there are often multiple applications that must consume that information, i.e., multiple data *sinks*, each one having its own data format, vocabulary, information requirements and protocol requirements. For example, the same source information may be used for patient care

purposes, research, quality-of-care measurement, billing, etc. This further complicates the data production process with more custom steps.

To automate the data production process when using semantic web technology, often an ad hoc pipeline is built using a mixture of shell scripts, SQL queries, SPARQL updates, web services, etc., and sometimes specialized integration tools. The resulting pipeline often uses a mix of interfaces ranging from files, to web services, HTTP, SQL, etc., and deals with a mixture of data representations such as text, CSV, XML and relational. On the plus side, such pipelines can be built using whatever tools are available for addressing each part of the problem, and the pipeline can evolve organically. On the minus side, such pipelines become extremely fragile, difficult to understand and difficult to maintain, both because they use so many technologies and because the topology of the pipeline is very hard for a newcomer to figure out. Typically, the topology is not expressed explicitly in one document – unless someone manually documented the pipeline, in which case the documentation is likely out of date. Instead, the topology is implicit in the communication that occurs between a shell script on one server, another shell script on another server, a web service on yet another server, etc. Furthermore, requirements frequently change as new data sources and new applications are integrated, thus causing pipeline maintenance to be a major problem.

To simplify the creation and maintenance of automated data production pipelines, various pipeline languages, tools and frameworks have been created over the years. For example, much research has already been done on workflow automation[2], and the W3C in 2010 standardized an XML pipe processing model, XProc[3]. Although the work presented in this paper could be considered workflow automation, it differs from most work in that area (and XProc) in that: (a) it is specifically oriented toward semantic web data production pipelines; and (b) it is more primitive, as there is no flow of control, no flow of control operators, and no central controller. A few other frameworks have been developed specifically for semantic web data production[4][5][6][7], but our work differs from those in being fully decentralized, with no central controller.

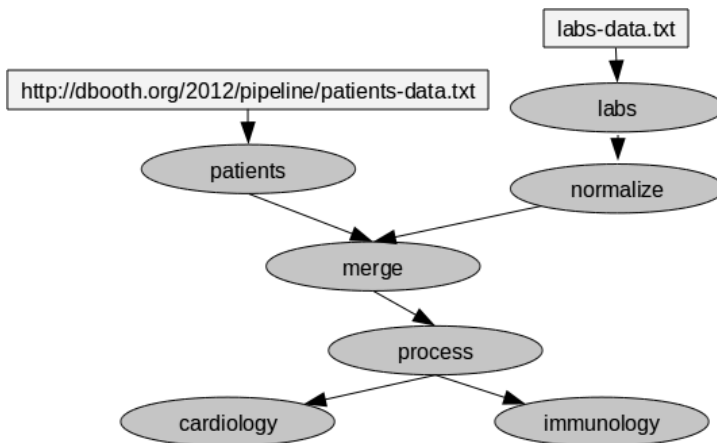
This paper presents an approach for semantic web data production pipelines that is unique in being decentralized – there is no central controller – distributed, web oriented (based on RESTful[8][9] HTTP), dependency graph driven, and allows adjacent nodes in a pipeline to transparently use local data-access methods when the nodes are compatible and on the same server. The approach was designed for semantic web applications but can also be used for other purposes. The approach has been implemented in the **RDF Pipeline Framework[10], an open source project available under the Apache 2.0 license**. The Framework provides: (a) a hosting environment (initially Apache2 using mod\_perl2) with a pluggable wrapper interface; and (b) some standard wrappers such as FileNode, GraphNode and JavaNode. (Wrappers are discussed in Section 3.1.) The user provides: (a) an RDF pipeline definition (such Pipeline #1 shown below); and (b) updaters (described below). As of this writing (4-May-2013), code for the RDF Pipeline Framework is in "developer

release" status: it runs and passes regression tests, and may be downloaded for testing, but is not yet ready for general production release, as some code cleanup and documentation still need to be done.

## 2 Example Pipeline

To provide a concrete basis for illustrating this approach, this section presents a simple example of a data pipeline using the RDF Pipeline Framework. Figure 1 shows a data pipeline (Pipeline 1) for producing cardiology and immunology data based on patient medical records and lab data. There is no special significance to the names of the nodes in this pipeline (patients, labs, normalize, merge, process, cardiology, immunology). They were chosen only to suggest the application-specific processing that they might perform.

To keep the example very simple, only two data sources are used and they are both text files, though one comes from a remote HTTP source and the other from a local file. (Of course, an actual system would likely involve more data sources and the data sources would often be things like relational databases or web services.)



**Fig. 1.** This simple data pipeline (**Pipeline 1**) shows patient and lab data being combined to produce data that is consumed for cardiology and immunology purposes. Each node in the graph performs arbitrary application-specific processing and data storage. A directed link from one node to another indicates data flow and hence data dependency. Although the lab data is related to the patient data – lab results for patients – the lab data is first run through a "normalize" step before being merged with the patient data. After merging, the data is further processed through another application specific step before being consumed by the cardiology and immunology nodes.

Here is the content from <http://dbooth.org/2012/patients-data.txt>:

```

patient id=001 name=Alice      dob=1979-01-23
patient id=002 name=Bob       dob=1950-12-21
patient id=003 name=Carol     dob=1944-06-12
  
```

```

patient id=004 name=Doug      dob=1949-08-27
patient id=005 name=Ellen    dob=1966-09-29
patient id=006 name=Frank    dob=1971-11-15

```

And here is the content of file **labs-data.txt**:

```

lab      customer=001    glucose=75      date=2012-02-01
lab      customer=002    glucose=85      date=2012-02-02
lab      customer=002    glucose=94      date=2012-02-03
lab      customer=004    glucose=72      date=2012-03-01
lab      customer=004    glucose=104     date=2012-03-02
lab      customer=004    glucose=95      date=2012-03-03
lab      customer=005    glucose=98      date=2012-02-02
lab      customer=006    glucose=87      date=2012-01-15
lab      customer=006    glucose=91      date=2012-01-16

```

The pipeline of Figure 1 (Pipeline 1) is defined in RDF/Turtle[11] as follows. Line numbers have been added for reference purposes.

```

1. # Pipeline 1: RDF/Turtle for Figure 1
2. @prefix p: <http://purl.org/pipeline/ont#> .
3. @prefix : <http://localhost/node/> .
4.
5. :patients a p:FileNode ;
6.   p:inputs ( <http://dbooth.org/2012/patients-data.txt> ) .
7.
8. :labs a p:FileNode ;
9.   p:inputs ( "labs-data.txt" ) .
10.
11. :normalize a p:FileNode ;
12.   p:inputs ( :labs ) .
13.
14. :merge a p:FileNode ;
15.   p:inputs ( :patients :normalize ) .
16.
17. :process a p:FileNode ;
18.   p:inputs ( :merge ) .
19.
20. :cardiology a p:FileNode ;
21.   p:inputs ( :process ) .
22.
23. :immunology a p:FileNode ;
24.   p:inputs ( :process ) .

```

It is easy to see that this pipeline definition corresponds directly to the graphical representation in Figure 1. Indeed, although Figure 1 was drawn manually, tools such as TopBraid Composer[12] can automatically display graphical representations of these pipelines, making them very easy to visualize. Some notes:

**Line 2:** Prefix "p:" is declared for the namespace <http://purl.org/pipeline/ont#> of the RDF Pipeline Framework's vocabulary. This is the vocabulary used to define a pipeline in the RDF Pipeline Framework, as summarized in Section 3.7.

**Line 3:** Prefix ":" is declared for the base URI `<http://localhost/node/>` of the RDF Pipeline server that will host one or more nodes in the pipeline. Any number of servers may be used, though this example uses only one.

**Line 5:** Node `<http://localhost/node/patients>` (abbreviated as `:patients`) is defined to be of type `p:FileNode`, which is the kind of wrapper (see Section 3.1) to be used by the `:patients` node. In web style, a node's URI is used both to identify that node and to retrieve data from it.

**Line 6:** The `:patients` node takes its input from a remote source, `<http://dbooth.org/2012/pipeline/patients-data.txt>`.

**Lines 11-12:** The `:normalize` node takes the result of the `:labs` node as its input.

**Lines 14-15:** The `:merge` node has two inputs, specified as an ordered list: the `:patients` node and the `:normalize` node.

Although Pipeline 1 defines the data flow between nodes, it supplies no details about the application-specific processing that is performed by each node. This separation of concerns makes it easy to reconfigure the pipeline without affecting the application-specific processing, and vice versa.

To specify the application-specific processing that a node should perform, an *updater* must be supplied. An *updater* is a named function, command or other operation that implements the processing task of a node. An updater is written by the user to perform an application-specific operation that produces data. Its job is to produce the node's output when invoked by its wrapper (described in Section 3.1). The wrapper passes, to the updater, wrapper-specific parameters for the node's inputs and output destination, such as filenames for a `FileNode`, or RDF graph names for a `GraphNode`.

For a node of type `p:FileNode`, such as `:patients`, the updater must be an executable program that accepts files as inputs and writes its output to stdout or (optionally) to a file. By default, the framework expects the name of the updater to be the node name implicitly, but it may also be specified explicitly using the `p:updater` property. Below is the updater for the `:patients` node, written as a shell script. Again, the line numbers are not a part of the script.

```
1. #! /bin/sh
2. # This is the patients node updater.
3. cat $1 | ./patients2rdf
```

This updater simply pipes the content of file `$1` through `./patients2rdf` and writes the result to stdout. Significant things to notice:

- There are no Application Programmer Interface (API) calls to pollute the updater code. Instead, the RDF Pipeline Framework invokes the updater when the data for that node needs to be generated, allowing the updater to be clean, simple and focused only on the application-specific task that it needs to perform.
- The updater expects its input as a file whose name is passed in as a parameter `$1` to the script, even though the pipeline definition specified its input as



<<http://dbooth.org/2012/patients-data.txt>>. The RDF Pipeline Framework will automatically cache – in a file – the content retrieved from <http://dbooth.org/2012/patients-data.txt>, and provide the cache filename as the actual parameter \$1 when it invokes the updater.

As shown in line 15 of Pipeline 1, the `:merge` node expects two inputs – the `:patients` node and the `:normalize` node. Here is the `:merge` updater.

```
1. #! /bin/sh
2. # This is the merge node updater.
3. cat $1
4. cat $2 | sed 's/customer/patient/g'
```

The `:merge` updater performs a crude RDF merge by concatenating files \$1 and \$2 to stdout. It also performs some crude ontology alignment by filtering file \$2 through `sed` in the process, to change all occurrences of "customer" to "patient", because the `:labs` data used the word "customer" where the `:patients` data used the word "patient". (Warning: this technique of using `cat` and `sed` to merge and edit RDF data will only work for certain kinds of data, and should not be used in general. It is shown here only to keep the example short and simple.)

Because the inputs of the `:merge` node are specified as an ordered list in the pipeline definition, parameter \$1 of the `:merge` node updater corresponds to the output of the `:patients` node, and parameter \$2 corresponds to the output of the `:normalize` node.

Once deployed, each node in a pipeline is independently "live", and will respond to data requests by dereferencing the node's URI. Thus, there are no specially designated endpoints: any node can be used as an endpoint or as an intermediate node. For example, if the `:patients` node is dereferenced – such as by pasting its URI into a browser, or by using the `curl[13]` command – the updater program named *patients* will be invoked (if necessary) and its output will be returned. Here is the output of "`curl http://localhost/node/patients`", with XSD data types[14] omitted for brevity:

```
@prefix patient: <http://example/patient#> .
@prefix : <http://example/med#> .
patient:p001 :lab [ :name "Alice" ; :dob "1979-01-23" ] .
patient:p002 :lab [ :name "Bob" ; :dob "1950-12-21" ] .
patient:p003 :lab [ :name "Carol" ; :dob "1944-06-12" ] .
patient:p004 :lab [ :name "Doug" ; :dob "1949-08-27" ] .
patient:p005 :lab [ :name "Ellen" ; :dob "1966-09-29" ] .
patient:p006 :lab [ :name "Frank" ; :dob "1971-11-15" ] .
```

And here is the output of "`curl http://localhost/node/merge`":

```
@prefix patient: <http://example/patient#> .
@prefix : <http://example/med#> .
patient:p001 :lab [ :name "Alice" ; :dob "1979-01-23" ] .
patient:p002 :lab [ :name "Bob" ; :dob "1950-12-21" ] .
```

```

patient:p003 :lab [ :name "Carol" ; :dob "1944-06-12" ] .
patient:p004 :lab [ :name "Doug" ; :dob "1949-08-27" ] .
patient:p005 :lab [ :name "Ellen" ; :dob "1966-09-29" ] .
patient:p006 :lab [ :name "Frank" ; :dob "1971-11-15" ] .
@prefix patient: <http://example/patient#> .
@prefix : <http://example/med#> .
patient:p001 :lab [ :glucose 750 ; :date "2012-02-01" ] .
patient:p002 :lab [ :glucose 850 ; :date "2012-02-02" ] .
patient:p002 :lab [ :glucose 940 ; :date "2012-02-03" ] .
patient:p004 :lab [ :glucose 720 ; :date "2012-03-01" ] .
patient:p004 :lab [ :glucose 1040 ; :date "2012-03-02" ] .
patient:p004 :lab [ :glucose 950 ; :date "2012-03-03" ] .
patient:p005 :lab [ :glucose 980 ; :date "2012-02-02" ] .
patient:p006 :lab [ :glucose 870 ; :date "2012-01-15" ] .
patient:p006 :lab [ :glucose 910 ; :date "2012-01-16" ] .

```

This technique of making each node independently "live" means that no central controller is needed or used, though nodes in a pipeline do share the same pipeline *definition*. It also allows the pipeline to be used for multiple applications that share some, but not all of the same data requirements. For example, the pipeline may have originally been built to supply an application with data from only the `:merge` node. The `:cardiology` and `:immunology` nodes may have been added later for other applications, without duplicating work or disrupting the existing pipeline. Furthermore, since each node can be on a different server (if desired), accessing its own private data, nodes can run concurrently.

The RDF Pipeline Framework does not currently check to see if a pipeline contains a cycle, although such a check would be straight-forward to add using well-known techniques. Since a pipeline definition indicates data dependencies, a cycle would likely be a mistake, though it is conceivable that a use could be found for it.

### 3 The RDF Pipeline Approach: What It Does and How It Works

This section describes more of the principles used in this approach, how they work and how they are used.

#### 3.1 Wrappers

Pipeline 1 above showed how an updater could be implemented by an arbitrary executable program, such as a shell script, which took files as inputs and produced a file as output. However, although shell scripts and files are convenient in many cases, data preparation for semantic web applications often requires processing steps that are more conveniently and efficiently performed directly within an RDF data store. Approaches like this are convenient for transforming RDF data from one model, ontology or vocabulary to another. For example, SPARQL 1.1 Update[15] operations can be used to create RDF named graphs from other named graphs. One can consider such tasks to be nodes in a pipeline, in which SPARQL Update operations take named graphs as inputs and produce named graphs as outputs.

To accommodate such needs, a node is composed of two parts: the updater and a *wrapper*. A *wrapper* is a standard component, usually provided by the Framework, that is responsible for invoking the updater and communicating with other nodes. This architecture allows updaters to be written in any programming language and consume or produce any kind of object, provided that a suitable wrapper is available. A wrapper runs inside a *hosting environment* that implements an HTTP server (e.g., Apache2/mod\_perl2 or Tomcat), allowing the wrappers to respond to HTTP requests, and in turn potentially invoking updaters. The wrapper framework is extensible, so new wrapper types can be plugged in to each hosting environment. The wrapper must be implemented in the same programming language as its hosting environment (e.g., Perl or Java), but this does not necessarily need to be the same language in which updaters are written – it depends on the wrapper.

Some basic wrappers:

- **p:FileNode**, for updaters written as executable programs (in any programming language) that consume and produce files;
- **p:GraphNode**, for updaters written as SPARQL Update operations that consume and produce RDF named graphs in a SPARQL server; and
- **p:JavaNode**, for updaters written in Java that consume and produce Java objects in a JVM.

For example, the following SPARQL Update code INSERTs presidents from graph `http://example/in` to graph `http://example/out` whose `foaf:givenName` is "Bill", changing the `foaf:givenName` to "William". Again, the line numbers are not a part of the code.

```

1. # SPARQL Updater #1
2. PREFIX foaf:      <http://xmlns.com/foaf/0.1/>
3. PREFIX inGraph:  <http://example/in>
4. PREFIX outGraph: <http://example/out>
5.
6. DROP SILENT GRAPH outGraph: ;
7.
8. INSERT {
9.   GRAPH outGraph: {
10.    ?president foaf:givenName "William" .
11.    ?president foaf:familyName ?familyName .
12.   }
13. }
14. WHERE {
15.   GRAPH inGraph: {
16.    ?president foaf:givenName "Bill" .
17.    ?president foaf:familyName ?familyName .
18.   }
19. }
```

Unfortunately, although the above code could be used as a `p:GraphNode` updater, it would not be very convenient or flexible, because the names of the input and output graphs are hard coded. Thus, the code would need to be modified if the pipeline were reconfigured to use a different input or output graph. It would be nice if the graph

names were instead passed in as parameters, so that this same SPARQL code could be used on any input and output graphs, but SPARQL 1.1 does not provide any way to do that. The RDF Pipeline Framework therefore includes a simple template facility that can be used for this purpose. Here is the same updater code, but written as a SPARQL Update template.

```

1. # SPARQL Updater #2, using a template
2. #inputs ( ${in} )
3. #outputs ( ${out} )
4.
5. PREFIX foaf:      <http://xmlns.com/foaf/0.1/>
6. PREFIX inGraph:  <${in}>
7. PREFIX outGraph: <${out}>
8.
9. DROP SILENT GRAPH outGraph: ;
10.
11. INSERT {
12.   GRAPH outGraph: {
13.     ?president foaf:givenName "William" .
14.     ?president foaf:familyName ?familyName .
15.   }
16. }
17. WHERE {
18.   GRAPH inGraph: {
19.     ?president foaf:givenName "Bill" .
20.     ?president foaf:familyName ?familyName .
21.   }
22. }
```

Points worth noting:

**Line 1** is a normal SPARQL comment line.

**Line 2** tells the SPARQL template processor the names of this updater's formal input parameters. When the template is expanded at runtime, this line will be removed and every occurrence of `${in}` will be changed to the URI of the node's input.

**Line 3** tells the SPARQL template processor the names of this updater's formal output parameters. When the template is expanded at runtime, this line will be removed and every occurrence of `${out}` will be changed to the node's URI.

### 3.2 Serializing, Deserializing and Optimizing Communication

In addition to invoking a node's updater, the wrapper is responsible for communication between nodes. Thus, the wrapper performs wrapper-specific serialization of node data (such as serializing a graph to RDF/Turtle) when it needs to transmit that data to an external node or other requester, and it performs the corresponding deserialization upon receiving data from an external node. This allows updaters to stay very simple – unpolluted by serialization, deserialization or data transmission issues.

This wrapper architecture also allows adjacent nodes to communicate more efficiently when they are on the same server and use the same wrapper. Instead of serializing an object, transmitting it via HTTP and deserializing it on receipt, nodes in the same environment can transparently access each other's objects directly. For example, if node `<http://example/in>` were an input to node `<http://example/out>` in a pipeline, and both nodes were `p:GraphNode`s in the same server, then the updater for `<http://example/out>` would automatically directly access the graph produced by `<http://example/in>`, avoiding both HTTP and serialization / deserialization.

Pipelines can be built from a heterogeneous mix of node types as long as the serializations produced by the wrappers are compatible. For example, a `p:FileNode` could produce output that is RDF/Turtle and be used as the input of a `p:GraphNode`.

### 3.3 Caching and Updating Only When Necessary

A wrapper does not necessarily invoke a node's updater for every data request. The wrapper automatically caches a node's output and keeps track of whether any of the node's inputs have changed. The updater is invoked only if the cached output is stale with respect to the nodes inputs. Again, this allows updaters stay simple, focusing only on the application-specific tasks that they need to perform.

### 3.4 Deploying and Distributed Processing

As of this writing, a pipeline is deployed by placing the pipeline definition file and updaters into the deployment directory of each hosting environment and starting the hosting environments, such as Apache2. However, a future version of the Framework will likely allow the pipeline definition to be read from an arbitrary HTTP source, thus simplifying the distribution of a new version of the pipeline definition to multiple hosting environments.

Nodes in a pipeline can be deployed on any servers that are accessible to their adjacent nodes. Consider the following simple two-node pipeline.

```

1. # Pipeline 2
2. @prefix p: <http://purl.org/pipeline/ont#> .
3. @prefix b: <http://server1.example.com/> .
4. @prefix w: <http://server1.example.com/> .
5. b:bills a p:GraphNode ;
6.   p:inputs ( <http://dbooth.org/2012/presidents.ttl> ) .
7. w:williams a p:GraphNode ;
8.   p:inputs ( b:bills ) .

```

Lines 3 and 4 of Pipeline 2 indicate that the `b:bills` and `w:williams` graphs are actually in the same SPARQL server (`server1.example.com`), and thus the `p:GraphNode` wrapper will cause the `w:williams` node to access `b:bills` graph directly. In contrast, if we had deployed these nodes on different servers (`server1.example.com` and `server2.example.com`) the pipeline definition would differ only on line 4, as shown in Pipeline 3 below. Furthermore, the updaters would not change at all.

```

1. # Pipeline 3
2. @prefix p: <http://purl.org/pipeline/ont#> .
3. @prefix b: <http://server1.example.com/> .
4. @prefix w: <http://server2.example.com/> .
5. b:bills a p:GraphNode ;
6.   p:inputs ( <http://dbooth.org/2012/presidents.ttl> ) .
7. w:williams a p:GraphNode ;
8.   p:inputs ( b:bills ) .

```

### 3.5 Update Policies

Consider Pipeline 3 above, and suppose that the data from node `b:bills` changes. When should the updater of node `w:williams` be invoked to update its output? Should it be updated immediately? Or should it be updated only when its output is actually requested? Or perhaps periodically, every  $n$  seconds?

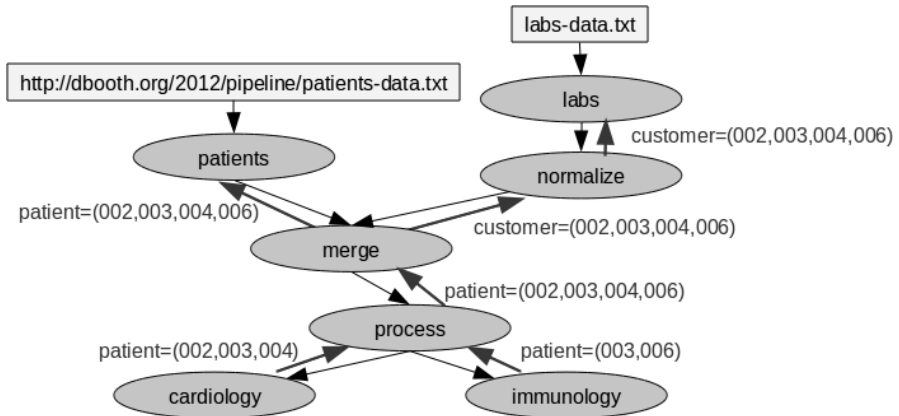
A node's `p:updatePolicy` may be specified as an additional node property to indicate the policy that the wrapper should use in deciding when to invoke a node's updater. Potential policies include `p:lazy`, `p:eager` and `p:periodic` – each one identifying a particular algorithm that will be used internally. Again, by specifying the update policy in the pipeline definition, a node's updater can stay simple.

### 3.6 Passing Parameters Upstream

Pipeline 1 above showed `:cardiology` and `:immunology` both consuming data. However, each one may only need a small subset of the total data that is available. It would be wasteful to propagate all possible `:patients` and `:labs` data through the pipeline if only a small subset is actually needed. For example, `cardiology` may only need data for `patient=(002,003,004)`, and `immunology` may only need data for `patient=(003,006)`.

To avoid this problem, parameters can be passed upstream through the pipeline, as illustrated in Figure 2. By default such parameters are passed as query string parameters on a node's URI, when node data is requested. For example, the command `"curl 'http://localhost/node/cardiology?patient=(002,003,004)'"` will request data from node `:cardiology`, passing query string `"patient=(002,003,004)"` as a parameter. (Of course, if parameters contain sensitive information then they should be suitably encrypted.) A parameter is treated as an additional node input, and thus a parameter change can cause the node's updater to fire. A node's updater can make use of its parameters if it chooses to do so. For example, for a `p:FileNode` updater, the most recently passed parameter is available in the `$QUERY_STRING` environment variable, and the parameters from all of a node's output nodes are available in the `$QUERY_STRINGS` environment variable.

By default, parameters are propagated upstream automatically. However, a pipeline definition may specify a `p:parametersFilter` for any node in order to transform the parameters as they are propagated upstream through that node. A `p:parametersFilter` is thus analogous to an updater, but it only operates on parameters



**Fig. 2.** Parameters are passed upstream through the pipeline, to control the data that the `:patients` and `:labs` nodes will generate. By default, parameters are passed upstream without modification. However, the pipeline definition may specify a `p:parametersFilter` for a node to control how that node will combine and/or modify the parameters that it passes upstream. In this illustration, the `:process` node supplied a `p:parametersFilter` that merged the parameters "patient=(002,003,004)" and "patient=(003,006)" that it received from `:cardiology` and `:immunology`, to produce the parameter "patient=(002,003,004,006)" that it passes upstream to the `:merge` node. The `:merge` node then used another `p:parametersFilter` to pass one parameter "patient=(002,003,004,006)" upstream to the `:patients` node, but a different parameter "customer=(002,003,004,006)" to the `:normalize` node. No other node in this pipeline needs to specify a `p:parametersFilter`. By passing such parameters upstream, the `:patients` and `:labs` updaters are able to generate only the data that is actually needed downstream.

that are being passed upstream. For a `p:FileNode`, the `p:parametersFilter` must be an executable program – typically a simple shell script. This treatment of parameter propagation again allows updaters to stay simple, while providing a powerful technique for data production to be efficiently controlled.

### 3.7 Error Checking and Automated Transformations

As of this writing, the Framework provides minimal error checking and does not include monitoring or alerting functions, though such features could be added in a future version. The Framework itself would be useful in implementing such features. For example, it would be easy to write an email notification node that reads from an error stream.

The Framework knows almost nothing about the semantics of a node or its inputs or output. It does not check to ensure that the actual input that a node receives conforms to the media type that the node expects, nor does the Framework perform any automatic transformation from one media type to another. It would be straightforward to extend the Framework to add such error detection and/or automatic transformation, but this has not been done thus far, because: (a) the user would have

to declare the expected media types for each of a node's inputs, thus making the pipeline definition more verbose; (b) the correspondence between the pipeline definition and the actual processing would be less direct, since in essence the Framework would perform automatic translation by inserting implicit translation nodes into the pipeline as needed; (c) a node normally has input expectations that go far beyond what a media type specifies, and during development these expectations need to be tested anyway, to ensure that the node receives what it expects, so it seems quite unlikely that a media type mismatch would pass unnoticed during such testing; and (d) it is very easy to insert an explicit translation node into a pipeline anyway.

Since an updater can perform arbitrary processing, updaters can have side effects that are unknown (and unknowable) to the Framework. Such side effects could cause concurrency issues if different updaters share the same resource. Users should bear this in mind when designing their updaters.

### 3.8 Graceful Evolution of Nodes and Pipelines

One motivation for cleanly separating the application-specific concerns (encapsulated in a pipeline's updaters) from the mechanics of caching, updater invocation, serialization, deserialization and handling HTTP requests, is to enable nodes and pipelines to evolve gracefully, without impacting other part of the pipeline: loose coupling. For example, a node can be swapped out for a new version, implemented in an entirely different programming language, with no change to adjacent nodes and only a trivial change to the pipeline definition (to change the node's wrapper type). This enables a pipeline to be developed quickly and easily, using the simplest available updater implementation techniques, and then refined as needed, adding features or improving efficiency. This fits well with agile development practices.

### 3.9 RDF Pipeline Properties

Section 3.1 discussed wrappers, which are represented in a pipeline description as classes. The following table summarizes the user-oriented properties used in defining a pipeline. Wrappers use additional properties internally. The subject (or domain) of each property in the table is a node unless the Value column indicates otherwise, such as "Subject is \$nodeType", which means that the subject should be the *type* of a node, e.g., `GraphNode`, rather than a node instance. For all properties (and classes) the namespace is `<http://purl.org/pipeline/ont#>` except for the `rdfs:type` property, `a/k/a "a"` in Turtle.

Property	Value
<code>a / rdfs:type</code>	Node type, e.g., <code>GraphNode</code> .
<code>contentType</code>	HTTP Content-Type for this node's serialized output. Defaults to <code>defaultContentType</code> of the <code>\$nodeType</code> .
<code>defaultContentType</code>	Subject is <code>\$nodeType</code> . Default HTTP Content-Type for serialized output.



defaultContentEncoding	Subject is \$nodeType. Default HTTP Content-Encoding for serialized output.
dependsOn	URIs of inputs, parameters and anything else this node depends on. Inputs and parameters are automatically included, but dependsOn can be used to specify additional dependencies.
hostRoot	Subject is \$nodeType. The value is a list that maps the server prefix (such as "http://localhost") of node URIs of this \$nodeType to the root location (as native name) of the server that implements the wrapper for this \$nodeType. Analogous to \$DOCUMENT_ROOT, which is used by default if this property is not set. Example: <pre>p:GraphNode p:hostRoot   ( "http://localhost" "http://localhost:28080/openrdf-workbench/repositories/owlimlite/" ) .</pre>
inputs	URIs of this node's inputs. They maybe other RDF Pipeline Nodes, or arbitrary HTTP data sources.
parametersFilter	File path of parametersFilter, relative to server "\$ENV{DOCUMENT_ROOT}/node/".
state	Native name of node output, i.e., the object that will be updated by the node's updater. For example, for a FileNode it is a filename. For a GraphNode it is a named graph.
stateType	Subject is \$nodeType. Type of state, if set. Otherwise \$nodeType is used.
stderr	File name of stderr from last update.
updatePolicy	Specifies the name of the algorithm that decides whether/when a node's state should be updated. Potential policies include lazy, eager and periodic.
updater	Native name of updater function.

## 4 Security

Data security is critical in healthcare and many other domains. For lack of space, this paper does not detail how security concerns can be addressed in the RDF Pipeline Framework, but as a brief outline:

- Wrappers can ensure that data in transit is securely encrypted, both in passing data downstream and in passing parameters upstream.
- Secure HTTP (https:) can also be used, for an additional layer of inter-node communication security.
- Updaters can ensure that data at rest is fully encrypted, if necessary.

## 5 Conclusions

This paper has presented a novel approach to automating data production pipelines for healthcare and other applications using semantic web technology. The approach makes use of framework-supplied *wrappers* that handle caching, dependency checking and inter-node communication, allowing a node's updater code to stay simple and application-focused. This also allows the framework to be used with multiple programming languages or object types, given appropriate wrappers. The approach is decentralized – every node in a pipeline is live – and nodes can be easily distributed across multiple servers with minimal change to the pipeline definition and no change to a node's updater. The approach is implemented as an open source project at <http://rdfpipeline.org/>. Interested parties are invited to contact the author.

## References

1. W3C: Resource Description Framework (RDF), <http://www.w3.org/RDF/> (retrieved June 08, 2012)
2. Anonymous, Articles on workflow (google scholar search), <http://tinyurl.com/a5yf5ng> (retrieved February 01, 2013)
3. Walsh, N., Milowski, A., Thompson, H., XProc: An XML Pipeline Language, W3C Recommendation (May 11, 2010), <http://www.w3.org/TR/xproc/> (retrieved February 01, 2013)
4. Becker, C., Bizer, C., Isele, R., Matteini, A., et al: Linked Data Integration Framework (LDIF), <http://www4.wiwiss.fu-berlin.de/bizer/ldif/> (retrieved June 08, 2012)
5. Top Quadrant: Sparql Motion, <http://www.topquadrant.com/products/SPARQLMotion.html> (retrieved June 08, 2012)
6. Phuoc, D.L., Morbidoni, C., Polleres, A., Samwald, M., Fuller, R., Tummarello, G.: DERI Pipes, <http://pipes.deri.org/> (retrieved June 08, 2012)
7. Fensel, D., van Harmelen, F., Witbrock, M., Carpentier, A.: LarkC: The Large Knowledge Collider, <http://www.larkc.eu/> (retrieved February 01, 2013)
8. Methedras: REST for the Rest of Us, [http://developer.mindtouch.com/REST/REST\\_for\\_the\\_Rest\\_of\\_Us](http://developer.mindtouch.com/REST/REST_for_the_Rest_of_Us) (retrieved June 08, 2012)
9. Fielding, R.: Chapter 5: Representational State Transfer (REST). From PhD Thesis: Architectural Styles and the Design of Network-based Software Architectures, University of California, Irvine (2000), [http://roy.gbiv.com/pubs/dissertation/rest\\_arch\\_style.htm](http://roy.gbiv.com/pubs/dissertation/rest_arch_style.htm) (retrieved June 08, 2012)
10. Booth, D.: rdf-pipeline, A framework for RDF data production pipelines, google code repository, <http://rdfpipeline.org/> (retrieved February 01, 2013)
11. Prud'hommeaux, E., Carothers, G. (eds.): Turtle: Terse RDF Triple Language (2011), <http://www.w3.org/TR/turtle/> (retrieved June 08, 2012)
12. TopQuadrant: TopBraid Composer, [http://www.topquadrant.com/products/TB\\_Composer.html](http://www.topquadrant.com/products/TB_Composer.html) (retrieved June 08, 2012)
13. Stenberg, D.: curl man page, <http://curl.haxx.se/docs/manpage.html> (retrieved June 08, 2012)
14. Biron, P.V., Malhotra, A.: XML Schema Part 2: Datatypes Second Edition (2004), <http://www.w3.org/TR/xmlschema-2/> (retrieved June 08, 2012)
15. Gearon, P., Passant, A., Polleres, A.: SPARQL 1.1 Update (2012), <http://www.w3.org/TR/sparql11-update/> (retrieved June 08, 2012)

# Towards Interoperable BioNLP Semantic Web Services Using the SADI Framework

Ahmad C. Bukhari, Artjom Klein, and Christopher J.O. Baker

Department of Computer Science and Applied Statistics,  
University of New Brunswick, Canada  
{sbukhari, aklein, bakerc}@unb.ca

**Abstract.** The number of NLP and BioNLP tools published as web services grows every year. Web services do not require installation, they are platform independent, and provide access to software modules that cannot be installed on regular computers due their complexity and heaviness. Whereas XML is the *de facto* interchange format for web services, the different XML schemas and the absence of semantics make the integration of resources (XML-based web services and their outputs) a very challenging task requiring significant effort from end users. We propose the use of *semantic web services* that provide semantic description of their in- and outputs to achieve interoperability of BioNLP services and the *ad-hoc* consolidation of their results. We leverage the SADI framework as a development platform to realize, by example, a number of highly integrated application and data integration scenarios.

**Project Page:** <https://code.google.com/p/bionlp-sadi>

**Keywords:** BioNLP, Bio data integration, Semantic web services, SPARQL.

## 1 Introduction

Over the last decade, biomedical natural language processing (BioNLP) has been validated as a solution to address the text-mining and information extraction needs of life scientists. Recently the number of NLP and BioNLP tools published as web services have been growing steadily. There are several providers of text-mining web services which include popular providers such as Whatizit [1], e-LICO [2], NaCTeM [3] and Manchester Interdisciplinary Biocentre [4]. Web services are typically registered in public catalogues (registries) e.g. BioCatalogue [5]. Web services do not require installation; they are platform independent and provide access to software that cannot be installed on desktop computers due to their complexity and heaviness. Most BioNLP tools produce XML based output, where XML schema represents syntactic structure of the input and output messages. In many use-cases, integration of several text-mining web services is required and the output results must be consolidated. Since XML schemas differ in their structure and do not provide the “meaning” of the

syntactic XML elements, integration of web services and consolidation of results cannot be automated and requires additional programming work.

Achieving semantic interoperability - where distinct data not only share the syntax (same structure) but also the same semantics (elements link to the same vocabulary providing system-independent interpretation of the data) is a bottleneck of XML-based approaches. Unlike XML-based web services, semantic web services provide semantic metadata describing their input and output. This enables automatic discovery, composition, interoperation, and ad-hoc consolidation of the outputs as long as they are modeled in terms of the same or compatible ontologies.

Despite the existence of several semantic web service specifications, such as OWL-S [6], SSWAP [7], WSMO [8], few have been successfully adopted. In our work we leverage the Semantic Automated Discovery and Integration (SADI) [9] framework. The choice is based on superior functionality of the framework for developing and deploying semantic web services, the availability of plug-in tools and client software that simplifies the discovery and utilization of the services by end users. Moreover, we have extensive experience using SADI to achieve semantic interoperability between data retrieval and data processing resources in several domains such as personalized medicine [10], clinical intelligence [11], ecotoxicology [12], lipidomics [13], mutation text-mining [14]. More details on SADI specification, tooling and usage see Methodology and Usage sections below.

The current goal of our work is to create a web based platform of interoperable services targeted at the bio text-miner, bioinformatics application or database developer where users can readily exploit service interoperability to perform complex tasks requiring ad-hoc mash-up of output data, without needing to program or install software. In this article we report on our work to leverage SADI as development platform to expose text-mining tools as SADI services and call our approach the *BioNLP-SADI* framework. We illustrate its utility for a number of scenarios of interest to life scientists. The article is outlined as follows. The Related Work describes the relevant approaches and shows the position of our approach in context of competing technologies. The Methodology section outlines the ontologies used for modeling the input and output of SADI services, as well as examples of service design and consolidated outputs. In the last section, we discuss possible use case scenarios and finally, the Conclusion summarizes the results and outlines directions of future work.

## 2 Related Work

Our work is inspired by the work described in [14]. The authors deployed a mutation impact extraction system as a SADI service in order to achieve interoperability between information extraction and data retrieving services. They also used a semantic client [15] that made it possible to build data processing pipelines from a semantic query, specifically for integrating mutation impact extraction text-mining with

pre-processing services (document retrieval, pdf-to-text conversion) and data visualization software which were also exposed as SADI services with compatible modeling. Using this client the creation of pipelines and consolidation of the outputs occurs in real time (on-the-fly) and does not require any integration effort by end users.

To the best of our best knowledge, there are no other attempts to expose BioNLP tools as semantic web services. Semantic web technologies have rarely been employed in the BioNLP domain. In a literature study, we found a modest number of attempts to use RDF/OWL to represent biomedical corpora [16] and biomedical text-mining results [17] [18]. One project in particular has adopted semantic web services to publish NLP tools, namely *nlp2rdf* [19]. This project aims to create an interchange format for linguistic tools, resources, and annotations. They developed ontologies to model basic document and text structure; a String Ontology [20] and a Structured Sentence Ontology [21]. They model Strings, Words, Sentences, their boundaries, and relations between them, such as *subString*, *superString*, *beginIndex*, *endIndex*, *rightContext*, *leftContext*, *nextSentence*, *previousSentence*, etc. The actual annotations in RDF are represented by using ontologies and vocabularies for a specific domain (syntactic parsing, part-of-speech tagging, named-entity-extraction, etc). A clear benefit is that once a NLP web service is modeled in terms of standard reference ontologies, its output can be published on the Web becoming Linked Open Data.

The general themes of interoperability, compatibility and re-usability of bio text-mining resources are currently being assessed by the BioCreative group through the organization and promotion of the BioCreative Interoperability Initiative [22]. This initiative aims to promote simplicity, interoperability, and the large-scale use and reuse of text mining resources by introducing and popularizing a new annotation standard – BioC, an interchange format for corpora and tools in BioNLP. The authors aim to achieve minimal interoperability - using basic categories such as sentences, tokens, parts of speeches, and several named entity categories. The work is at a very early stage and currently no detailed specification of the approach is available.

### 3 Methodology

In order to achieve semantic interoperability of text-mining web services and their outputs we propose to use a semantic web services framework, SADI in particular, and use reference ontologies to provide the compatible modeling. In the following we describe SADI technology, ontologies we use and give examples of service design and consolidated output automatically produced by example web services.

#### 3.1 SADI

The SADI framework [9] is a set of conventions for creating HTTP-based semantic web services that can be automatically discovered and orchestrated. SADI services consume RDF documents as input and produce RDF documents as output, which solves the syntactic interoperability problem. This is also convenient for client

programs that can leverage existing APIs for RDF to represent the data on which SADI services operate. In brief, SADI services operate by attaching new properties to input URIs described in the input RDF graph, and the most important feature of SADI is that these properties are fixed for each service. A declaration of these predicates, available online, constitutes a semantic description of the service. For example, if a service is declared with the predicate `hasPartOfSpeechAnnotation` described in ontology as a predicate linking document with a part-of-speech tagger service, the client software knows that it can call the service to generate part-of-speech annotations. The SADI framework uses the OWL class syntax to specify conditions on the input nodes a service consumes, and declare the predicates the service attaches. Such declarations of inputs and outputs of services enable completely automatic discovery and composition of SADI services. For more technical details on SADI and SHARE (<http://sadiframework.org/content/tag/share/>), the reader is referred to [9] [15].

### 3.2 Ontologies for Modeling Annotations and Extracted Entities

The interoperability of services and the ad-hoc mash-up of text-mining results can be achieved by defining text-mining web services output in terms of the same or compatible reference ontology(s). *BioNLP-SADI* leverages two types of ontologies: (1) ontologies for modeling annotations, and (2) domain ontologies for modeling entities and their relations extracted from text.

**Annotation Ontologies:** We propose to use the Annotation Ontology (AO) [23] to model the structural level of annotations. AO is an open-source ontology for annotating the scientific documents on the web. In AO, all the annotations are regarded as resources and fall under the instance category of the *Annotation* class. Each annotation has some *hasTopic* and *context* predicates and the object of *hasTopic* predicate can be a certain entity such as drug, chemical, disease, or reified fact while *context* refers to a certain text inside the sentence. This simple reference model makes it possible to connect extracted information to surface text or entire documents. The provenance of annotations is modeled with Provenance, Authoring and Versioning (PAV) ontology [24]. E.g. predicates such as *createdBy*, *createdOn* describe the creator and date of creation.

**Domain ontologies:** Domain ontologies are used to model information extracted from a document. Such ontologies can be an upper level ontology such as Semantic Integration Ontology (SIO) [25] which also includes many high level biomedical concepts, or any specific domain ontology.

### 3.3 IO Service Modelling Examples

Fig. 1 displays a prototypical model of the in- and output of a sentence splitting SADI service. The goal of the SentenceSplitter service is to split text into sentences. The service is represented by modeling the relation between the document and annotations

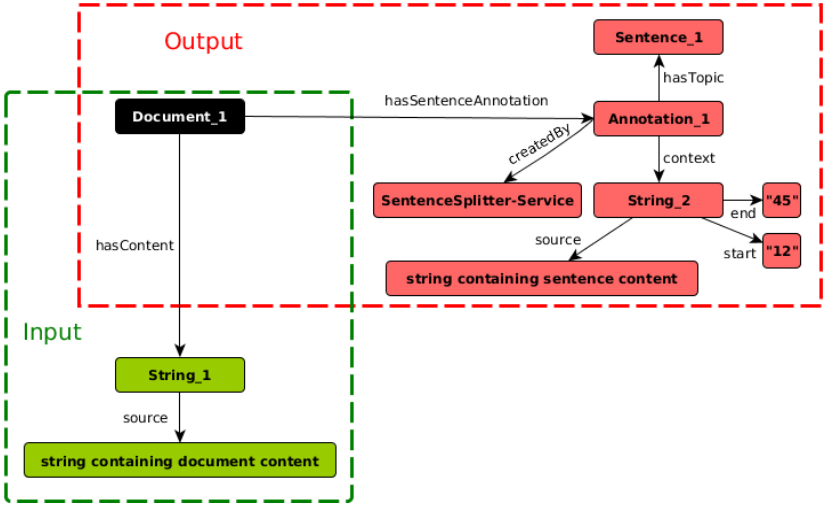


Fig. 1. A SADI service architecture of a sentence splitter service

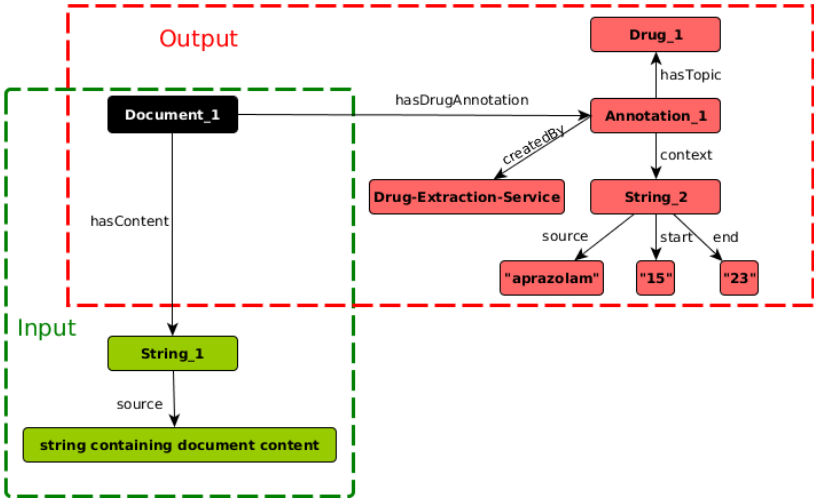


Fig. 2. A SADI service architecture of a drug extraction service

of type *Sentence* in text segments. It consumes an instance of *Document* class as input with the attached string content via the *hascontent* relation. Since most of the text mining services carry out the document annotation, they all share the same input modeling (compare with Fig. 2 that shows another text-mining service). In the output, the service attaches an *Annotation* instance via the predicate *hasSentenceAnnotation*. Typically the name of the property would represent the functionality of the services,

or type of annotations created by it. Accordingly, the service extracting drugs attaches *hasDrugAnnotation* predicates to the document (see Fig. 2.) Note the input and output entity is the same entity - a central feature of the SADI specification. The service simply decorates the input node (document) with new information (annotations). Annotations are characterized by their type - *Sentence* (Fig. 1) and *Drug* (Fig. 2) - and text strings with boundaries - *start* and *end*. Each service attaches provenance information about the creator (the service itself) and the time when annotations are created.

### 3.4 Consolidated Service Outputs

Using SADI as a platform guarantees the results of text-mining are interoperable with SADI services which operate on structured data. These data retrieval services can augment the results of services serving text extracted information. This can take the form of adding new data, on verifying text extracted results against existing data in knowledge bases and controlled vocabularies, a process known as grounding (assuming that knowledge bases have compatible semantic interface).

Fig. 3 shows a (prototypical) RDF graph automatically assembled from the merged outputs of three services. Document\_1 was annotated with drugs by the Drug-Extraction-Service and was split into sentences by the Sentence-Splitter-Service. Following this the Drug-Drug-Interaction-Service, which found all drug pairs with potentially harmful interactions. The Drug-Drug-Interaction service in our example is a

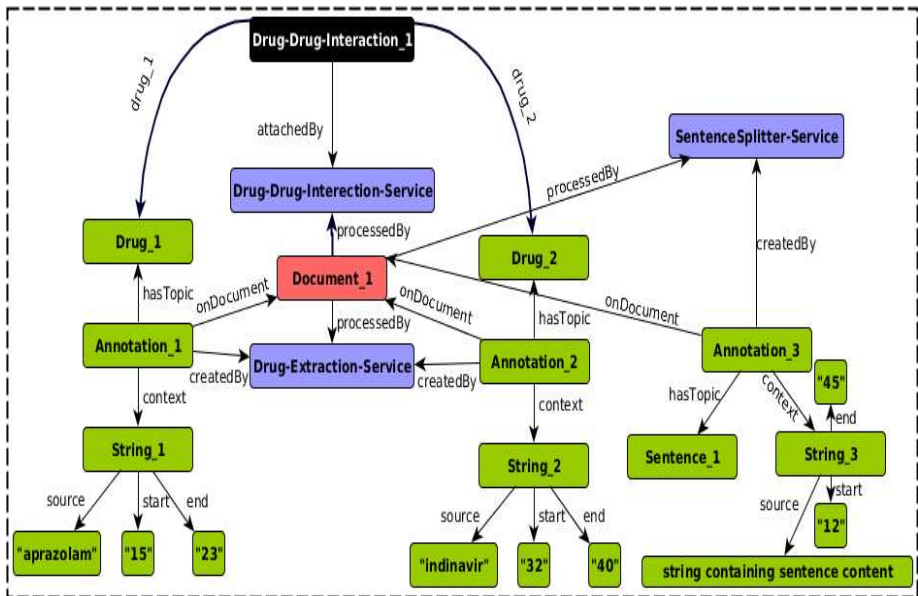


Fig. 3. Drug-Extraction and Drug-Drug-Interaction SADI services Consolidated Output



data-retrieval service (not a text mining service). It is based on the DrugBank [26] database that contains information about drugs and their interactions. The service retrieves for each drug all the known interactions from the database and attaches them to the drug as instances of the *Drug-Drug-Interaction* class (see Figure 3). The resulting output of three services makes it possible to pose queries for the target information such as, find all sentences where potentially harmful drug-drug interactions are mentioned.

## 4 Access to BioNLP-SADI Services

In this section we outline how to access interoperable BioNLP-SADI services. Firstly it is possible to interact with these services through a web based interface similar to the nlp2rdf web interface [27] where the users can select a service or combination of services to achieve a certain goal and provide input in the form of a text string or a list of Pubmed ids - similar to the Whatizit web interface [28]. Unlike XML-based services, our services will produce RDF with automatically mashed-up annotations and the RDF data will be available for download and be searchable via SPARQL interface.

For end users that are more familiar with semantics technologies it is possible to use a SADI SPARQL client like SHARE, which is a proof-of-concept Semantic Web query engine that resolves SPARQL (an RDF Query Language [29]) queries by building and executing workflows with SADI services. SHARE is also capable of discovering instances of a given OWL class by building an appropriate SADI workflow. Further details about SHARE can be read in [15]. The SADI client application can also be integrated into any NLP framework such as GATE [30] or UIMA [31]. Since we use the Annotation Ontology to model structural level of annotations, services can be also integrated into the DOME0 [32] graphical annotation toolkit, which is RDF-based and uses the same ontology to model annotations.

SADI services can be accessed via SADI API; moreover a SADI plugin for Taverna [33] will allow easy assembly of text mining pipelines. SADI services themselves can be registered and discovered in a SADI registry which has a web interface where service providers can register their services. The registry can also be accessed programmatically via SADI Registry API, or as a regular SPARQL endpoint. Finally, from developer's point of view, there is a rich infrastructure for developing and testing SADI services: service and client APIs, automatic service generator, Protege Plugin and etc.

## 5 Use Cases - Sample Queries

In this section, we focus on the type of information that could be extracted from ad-hoc consolidated outputs. As we outlined in the previous section, we can easily merge

the RDF outputs (because of the RDF interoperability attribute) produced by different SADI services to query the desired information. Here we present some use case scenarios along with simplified SPARQL queries and the already deployed SADI services to give an overview about ad-hoc queries to the user.

### **Use Case 1: Find abstracts where the same mutation mention is found in two adjacent sentences.**

In this scenario, a bioinformatics database curator wants to find abstracts where the same mutation occurs in adjacent sentences. The generalized SPARQL query below could extract the mutation in two adjacent sentences.

#### **SADI services employed in use case 1: Mutation Finder and Sentence Splitter.**

```
SELECT DISTINCT ?sentence
WHERE{
  ?document a Document; hasId ?doc_id .
  ?mutation a Mutation .
  ?annotation_1 a Annotation;
    hasTopic ?mutation;
    onDocument ?document;
    context
    [
      a String;
      source "N30A";
      isContainedBy ?sentence_1
    ] .
  ?annotation_2 a Annotation;
    hasTopic ?mutation;
    onDocument ?document;
    context
    [
      a String;
      source "N30A" ;
      isContainedBy ?sentence_2
    ] .
  ?sentence_1 hasNextSentence ?sentence_2 .
}
```

### **Use Case 2: Find Sentences Where Mutation and Drug Occur in the Same Sentence.**

In this use case, we address the needs of researchers involved in small molecule drug discovery seeking to retrieve sentences where mentions of mutations to drug targets and the small molecules are found in the same sentence. This is possible by invoking mutation finder and drug extraction services run in parallel (as they are not dependent on each other) and combining their results to generate a semantically enriched interoperable consolidated output view.

## SADI Services Employed in Use Case 2: Mutation Finder, Drug Extractor and Sentence Splitter

```

SELECT DISTINCT ?sentence
WHERE{
  ?document a Document;
    hasId ?doc_id .
  ?annotation_1 a Annotation;
    hasTopic [ a Mutation ];
    onDocument ?document;
    context
      [
        a String;
        source "N30A";
        isContainedBy ?sentence
      ] .
  ?annotation_2 a Annotation;
    hasTopic [ a Drug ];
    onDocument ?document;
    context
      [
        a String;
        source "Indinavir";
        isContainedBy ?sentence
      ] .
}

```

## Use Case 3: Extract Drug Mentions from Text and Display Known Interactions Between Them.

This query retrieves evidence of interactions between different drugs mentioned in a document and would be of interest to physicians looking for side effects (positive, negative, neutral) of a particular drug combination when given together to patients. The following SPARQL query identifies evidence of interaction between two drugs.

## SADI Services Employed in Use Case 3: Drug Extractor and Drug Drug Interaction

```

SELECT DISTINCT ?drug_1 ?drug_2
WHERE{
  ?document a Document;
    hasId ?doc_id;
    ?drug_1 a Drug .
  ?annotation_1 a Annotation;
    hasTopic ?drug_1;
    onDocument ?document .
  OPTIONAL {
    ?drug_2 a Drug .
    ?drug_1 interactsWith ?drug_2 .
  }
}

```

#### Use Case 4: Find Foods with Known Interactions with the Drug Cytarabine.

In this scenario, we aim to address the needs of both physicians and patients interested to know the interaction of any foods with a certain type of drug. Certain foods are known to be effective in reducing the effectiveness of a prescribed drug. The following SPARQL query can fetch all the food interactions with drug the Cytarabine.

#### SADI Services Employed in Use Case 4: Drug Extractor, Drug Food Interaction

```
SELECT DISTINCT ?food
WHERE{
    ?document a Document; hasId ?doc_id .
    ?annotation_1 a Annotation;
        hasTopic
        [
            a Drug;
            hasDrugBankId "DB00987"; # DrugBankID for Cytarabine
        ];
        onDocument ?document .
    ?annotation_2 a Annotation;
        hasTopic [ a Food ];
        onDocument ?document .
    ?drug hasDangerousInteractionWith ?food .
}
```

## 6 Conclusion

To address the ongoing challenges of integration among XML-based BioNLP web services, we proposed a generalized architecture for text mining web services using the SADI semantic web service framework. The sophisticated mechanism we propose is able to address key challenges related to BioNLP interoperability and data provenance. We have created in-house implementations of the services described here-in (drug-extraction, mutation mention detection, sentence-splitting, drug-drug-interaction) along with several SPARQL queries for use with a SHARE like client. The combined RDF outputs of the service calls permit the construction of more elaborate queries, matching the needs of our target end users. Although the work we have outlined is at an early stage, it has shown that we can provide new functionality and seamless interoperation between services, facilitating meaningful knowledge discovery. In the future work, we plan to integrate more BioNLP tools and develop more complex use cases based on combining interoperable text mining and data mining web services.

## References

1. Rebholz-Schuhmann, D., Gaudan, A.M., Kirsch, H., Jimeno, A.: Text processing through Web services: calling Whatizit. *Bioinformatics* 24(2), 296–298 (2008)
2. An e-Laboratory for Interdisciplinary Collaborative Research in Data Mining and Data-Intensive Science, <http://www.e-lico.eu/>

3. National Centre for Text Mining, <http://www.nactem.ac.uk/>
4. Manchester Institute of Biotechnology, <http://www.mib.ac.uk/>
5. The Life Science web services registry, <http://www.biocatalogue.org/>
6. OWL-S: Semantic Markup for Web Services, <http://www.w3.org/Submission/OWL-S/>
7. Damian, G., Schiltz, G., May, G., Avraham, S., Town, C., Grant, D., Nelson, R.: SSWAP: A Simple Semantic Web Architecture and Protocol for semantic web services. *BMC Bioinformatics* 10, 309 (2009)
8. Web Service Modeling Ontology (WSMO), <http://www.w3.org/Submission/WSMO/>
9. Wilkinson, M., Vandervalk, B., McCarthy, L.: The Semantic Automated Discovery and Integration (SADI) Web service Design-Pattern, API and Reference Implementation. *Journal of Biomedical Semantics* 2(1), 5–23 (2011)
10. Vandervalk, B., McCarthy, L., Toledo, J., Klein, A., Baker, C., Dumontier, M., Wilkinson, M.: The SADI Personal Health Lens: A Web Browser-Based System for Identifying Personally Relevant Drug Interactions. *JMIR Res. Protoc.* 2(1), e14 (2013)
11. Riazanov, A., Klein, A., Nejad, A., Rose, G., Forster, A., Buckeridge, D., Baker, C.: Semantic querying of relational data for clinical intelligence: a semantic web services-based approach. *J. Biomedical Semantics* 4, 9 (2013)
12. Riazanov, A., Hindle, M., Goudreau, E., Martyniuk, C., Baker, C.: Ecotoxicology Data Federation with SADI Semantic Web Services. *SWAT4LS* (2012)
13. Chepelev, L., Riazanov, A., Kouznetsov, A., Low, H., Dumontier, M., Baker, C.: Prototype semantic infrastructure for automated small molecule classification and annotation in lipidomics. *BMC Bioinformatics* 12(1), 303 (2011)
14. Riazanov, A., Laurila, J.B., Baker, C.: Deploying mutation impact text-mining software with the SADI Semantic Web Services framework. *BMC Bioinformatics* 2(4), 1471–2105 (2011)
15. Wilkinson, M., McCarthy, L., Vandervalk, B., Withers, D., Kawas, E., Samadian, S.: SADI, SHARE, and the in silico scientific method. *BMC Bioinformatics* 11(12), S7 (2012)
16. Croset, S., Grabmüller, C., Li, C., Kavaliauskas, S., Dietrich, R.: The CALBC RDF Triple Store: retrieval over large literature content. *CoRR*, 1012, 1650 (2012)
17. Naderi, N., Witte, R.: Automated extraction and semantic analysis of mutation impacts from the biomedical literature. *BMC Genomics* 13(4), S10 (2012)
18. Laurila, J., Naderi, N., Witte, R., Riazanov, A., Kouznetsov, A., Baker, C.: Algorithms and semantic infrastructure for mutation impact extraction and grounding. *BMC Genomics* 11(4), s24 (2011)
19. Sebastian, H., Jens, L., Sören, A.: NIF: An ontology-based and linked-data-aware NLP Interchange Format. In: *5th Workshop on Linked Data on the Web* (2012)
20. Sebastian, H., Lehmann, J., Auer, S.: Towards an ontology for representing strings. In: *Proceedings of the EKAW* (2012)
21. The Structured Sentence Ontology, <http://nlp2rdf.lod2.eu/schema/sso/>
22. BioCreative: Critical Assessment of Information Extraction in Biology, <http://www.biocreative.org>
23. Ciccarese, P., Ocana, M., Castro, L., Das, S., Clark, T.: An Open Annotation Ontology for Science on Web 3.0. *J. Biomed. Semantics* 2(2), S4 (2011)
24. Ciccarese, P., Wu, E., Wong, G., Ocana, M., Kinoshita, J., Ruttenberg, A., Clark, T.: The SWAN biomedical discourse ontology. *J. Biomed. Inform.* 41(5), 739–751 (2008)
25. The SemanticScience Integrated Ontology (SIO), <http://semanticscience.org/ontology/sio.owl>.

26. The DrugBank database, <http://www.drugbank.ca/>
27. NIF Combinator: Combining NLP Tool Output, <http://nlp2rdf.lod2.eu/demo.php>
28. EBI's Whatizit service, <http://www.ebi.ac.uk/webservices/whatizit/info.jsf>
29. SPARQL 1.1 Query Language, <http://www.w3.org/TR/sparql11-query/>
30. GATE: a full-lifecycle open source solution for text processing, <http://gate.ac.uk/>
31. The Unstructured Information Management Architecture (UIMA) framework, <http://uima-framework.sourceforge.net/>
32. Ciccarese, P., Clark, O.: Open semantic annotation of scientific publications using DOMEQ. *J. Biomed Semantics* 24(suppl. 3) (2012)
33. 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, 729–732 (2006)

# Optimizing Similarity Computations for Ontology Matching - Experiences from GOMMA

Michael Hartung<sup>1,2</sup>, Lars Kolb<sup>1</sup>, Anika Groß<sup>1,2</sup>, and Erhard Rahm<sup>1,2</sup>

<sup>1</sup> Department of Computer Science, University of Leipzig

<sup>2</sup> Interdisciplinary Center for Bioinformatics, University of Leipzig  
{hartung,kolb,gross,rahm}@informatik.uni-leipzig.de

**Abstract.** An efficient computation of ontology mappings requires optimized algorithms and significant computing resources especially for large life science ontologies. We describe how we optimized n-gram matching for computing the similarity of concept names and synonyms in our match system GOMMA. Furthermore, we outline how to enable a highly parallel string matching on Graphical Processing Units (GPU). The evaluation on the OAEI LargeBio match task demonstrates the high effectiveness of the proposed optimizations and that the use of GPUs in addition to standard processors enables significant performance improvements.

**Keywords:** ontology matching, GPU, parallel hardware.

## 1 Introduction

Mappings (alignments) between ontologies are important for many life science applications and are increasingly provided in platforms such as BioPortal [13]. New mappings are typically determined semi-automatically with the help of ontology match systems such as GOMMA (Generic Ontology Matching and Mapping Management) [10] utilizing different matchers to evaluate the linguistic and structural similarity of concepts [3]. Ontology matching is challenging especially for large ontologies w.r.t. both effectiveness (achieving a high quality mapping) and efficiency, i.e., fast computation [16]. Results of the 2012 OAEI [14] LargeBio task<sup>1</sup> showed that some systems still have problems or are even unable to match large ontologies such as the Foundation Model of Anatomy (FMA) [5] or the Thesaurus of the National Cancer Institute (NCIT) [11].

For high efficiency, it is important to reduce the search space by avoiding the comparison of dissimilar concepts [2,9], and to utilize optimized implementations for frequently applied similarity functions such as n-gram, Jaccard, TF-IDF (e.g., by using fast set intersection [1], or pruning techniques [18]). Libraries such as SimMetrics<sup>2</sup> typically provide a comfortable and general interface `getSim(string1, string2)` for multiple similarity measures but often lack efficient implementations. For example, they either lack pre-processing steps to transform strings

---

<sup>1</sup> <http://www.cs.ox.ac.uk/isg/projects/SEALS/oei/2012/>

<sup>2</sup> <http://sourceforge.net/projects/simmetrics/>

into representations permitting faster comparisons or they cause redundant pre-processing steps when matching a particular string multiple times.

A promising direction to speed-up processing-intensive computations such as string comparisons is the utilization of Graphical Processing Units (GPU) supporting a massively parallel processing even on low-cost graphic cards. The availability of frameworks like CUDA and OpenCL further stimulated the interest in general purpose computation on GPUs [15]. Algorithms like BLAST [17], database joins [8] or duplicate detection/link discovery systems [4,12] have already been adapted for GPU execution. Unfortunately, GPUs and their programming languages like OpenCL have several limitations. For instance, only basic data types can be used, the memory capacity of a GPU is restricted to a specific size, and data must first be transferred to the GPU. Furthermore, no dynamic memory allocation is possible, i.e., the resources required by an algorithm must be known and allocated a priori. These restrictions need to be considered in a new solution for computing string similarity in match systems such as GOMMA.

In this experience paper, we make the following contributions:

- We describe how we optimized n-gram matching for linguistic matching in GOMMA including the use of integer representations for n-grams. (Sec. 2)
- We propose a new method for n-gram matching on GPU. The technique is not limited to n-gram and can also be applied to other token-based string metrics (e.g., Jaccard). We further describe how GOMMA can exploit both CPU and GPU resources for improved efficiency. (Sec. 3)
- We evaluate our techniques on a real-world match problem, namely the FMA-NCI match task from the OAEI LargeBio track. The results show that we are able to significantly reduce the execution times on CPU as well as GPU compared to the standard solution. (Sec. 4)

## 2 Optimizing N-Gram Similarity Computation

GOMMA uses the n-gram string comparison to determine the similarity of names and synonyms for pairs of concepts of two input ontologies  $O$  and  $O'$ . The example in Fig. 1 shows the names/synonyms for three concepts per ontology. The match result is a mapping  $M_{O,O'} = \{(c, c', sim) \mid c \in O, c' \in O', sim \in [0, 1]\}$  consisting of correspondences between concepts and their match similarity. Given that a concept has a name and potentially several synonyms, there can be several n-gram similarities per pair of concepts. GOMMA thus applies an aggregation function **agg**, e.g., maximum or average, to aggregate multiple similarity values. Finally, GOMMA uses a threshold  $t$  to restrict the mapping to the most likely correspondences.

A naive n-gram matcher first splits the string attribute values to be compared into overlapping tokens of length  $n$ . For our example and  $n=3$  (Trigram), the  $c_2$  strings *limbs* and *extremity* are split into  $\{\{lim,imb\}, \{ext,xtr,tre,rem,emi,mit,ity\}\}$  while the single  $c'_2$  attribute value *limbs* is tokenized into  $\{\{lim,imb,mbs\}\}$ . To determine the similarity between two concepts, we (1) need to compute the dice



		O				O'			
		Attr. Values		Sorted Token Vectors		Attr. Values		Sorted Token Vectors	
Concept	$c_0$	• head	• [1,2]	• head	• [1,2]	$c_0'$			
	$c_1$	• trunk • torso	• [3,4,5] • [6,7,8]	• torso • truncus	• [6,7,8] • [3,4,18,19,20]	$c_1'$			
	$c_2$	• limb • extremity	• [9,10] • [11,12,13,14,15,16,17]	• limbs	• [9,10,21]	$c_2'$			

Dictionary		O'										
		hea	ead	tru	run	unk	tor	ors	rso	lim	imb	ext
1	2	3	4	5	6	7	8	9	10	11		
		xtr	tre	rem	emi	mit	ity	unc	ncu	cus	mbs	
12	13	14	15	16	17	18	19	20	21			

$M_{O,O'}$		O'			
		$c_0'$	$c_1'$	$c_2'$	
O	$c_0$	1.0	0.0	0.0	0.0
	$c_1$	0.0	0.0	4/9	0.0
	$c_2$	0.0	0.0	0.0	4/5
		0.0	0.0	0.0	0.0

**Fig. 1.** Example trigram similarity computation. Attribute values (names, synonyms) are converted to sorted token vectors (upper part). Tokens are represented as integers based on a dictionary (lower left part). Individual similarities are aggregated with the **max** function to determine an overall similarity between two concepts (lower right part).

coefficient for each pair of token sets  $TS_1-TS_2$  ( $\text{diceSim}(TS_1, TS_2) = \frac{2 \cdot |TS_1 \cap TS_2|}{|TS_1| + |TS_2|}$ ) and (2) aggregate the single similarities. For instance, when determining the similarity between  $c_2$  and  $c_2'$ , we compute two single similarities  $\text{diceSim}(\{\text{lim}, \text{imb}\}, \{\text{lim}, \text{imb}, \text{mbs}\}) = 0.8$  and  $\text{diceSim}(\{\text{ext}, \text{xtr}, \text{tre}, \text{rem}, \text{emi}, \text{mit}, \text{ity}\}, \{\text{lim}, \text{imb}, \text{mbs}\}) = 0$  which are aggregated using the **max** function:  $\text{sim}(c_2, c_2') = \max(0.8, 0) = 0.8$ .

There are several possibilities to compute the set intersection ( $TS_1 \cap TS_2$ ) that can have a large impact on efficiency, e.g., a nested loop over both element sets or the use of hash tables. Furthermore, in case of string-valued tokens, even the check whether two tokens (strings) are equal is a quite complex operation. In general, the larger the sets and the longer the tokens are, the more time is required to compute set overlaps. Since such similarity computations frequently occur in match workflows for a large number of concepts, it turns out that an efficient implementation of the token set intersection is a key factor to speed up the matching of large ontologies. In recent years, different optimization techniques for set similarity joins (e.g., prefix, suffix, and length filtering) were proposed in the context of near duplicate detection [18]. We omit those orthogonal optimizations in favor of readability and leave their application for future work.

GOMMA's optimized n-gram matcher is described in Algorithm 1. It is based on two optimizations: the use of integer representations for tokens and a sort-merge-based computation of token set overlaps. As in the naive approach, we first split a concept's string attribute values into token sets (Line 6). We then convert all n-grams into integer values based on a global dictionary (Line 8). The dictionary is built dynamically, i.e., each time a new n-gram is observed during the tokenization, it is added to the dictionary and, from then on, represented by its integer-valued index in the dictionary. Additionally, we sort the integer

---

**Algorithm 1.** `ngramSim( $O, O', Attr, agg, t$ )`

---

```

1  foreach  $c \in O \cup O'$  do
2     $c.stvs \leftarrow \emptyset$ ; // sorted token vectors
3     $S \leftarrow c.getAttrValues(Attr)$ ;
4    foreach  $s \in S$  do
5       $stv \leftarrow []$ ; // empty token vector
6       $tokens \leftarrow tokenizeNGrams(s)$ ;
7      foreach  $t \in tokens$  do
8         $n \leftarrow getNumericTokenId(t)$ ;
9         $stv.append(n)$ ;
10      $c.stvs \leftarrow c.stvs \cup \{stv.sort()\}$ ;
11   $M \leftarrow \emptyset$ ;
12  foreach  $c \in O$  do
13    foreach  $c' \in O'$  do
14       $Sims \leftarrow \emptyset$ ;
15      foreach  $stv \in c.stvs$  do
16        foreach  $stv' \in c'.stvs$  do
17           $s \leftarrow diceSim(stv, stv')$ ;
18           $Sims \leftarrow Sims \cup \{s\}$ ;
19       $sim \leftarrow agg(Sims)$ ;
20      if  $sim \geq t$  then
21         $M \leftarrow M \cup \{(c, c', sim)\}$ ;
22  return  $M$ ;

```

---



---

**Algorithm 2.** `diceSim( $stv_1, stv_2$ )`

---

```

1   $left \leftarrow 0$ ;
2   $right \leftarrow 0$ ;
3   $overlap \leftarrow 0$ ;
4   $l_1 \leftarrow stv_1.length()$ ;
5   $l_2 \leftarrow stv_2.length()$ ;
6  while  $(left < l_1) \wedge (right < l_2)$  do
7    if  $stv_1[left] == stv_2[right]$  then
8       $overlap++$ ;
9       $left++$ ;
10      $right++$ ;
11   else if  $stv_1[left] < stv_2[right]$ 
12     then
13        $left++$ ;
14     else
15        $right++$ ;
15  return  $2 \cdot overlap / (l_1 + l_2)$ ;

```

---

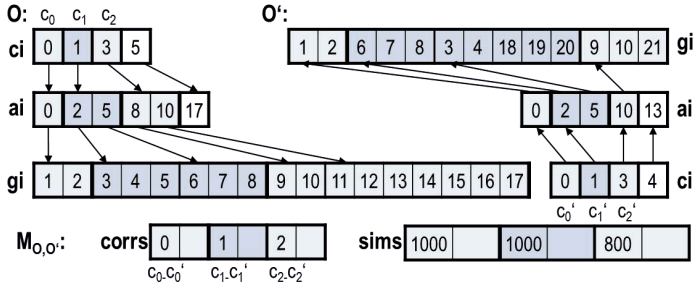
values of each token vector in ascending order (Line 10). Thus, after this pre-processing, a concept has a set of sorted token vectors (*stvs*) representing the n-grams of their string attribute values as integers. For example, the trigrams of  $c_2$  are represented as  $\{[9,10], [11,12,13,14,15,16,17]\}$ .

We then iterate over all concepts of  $O$  and  $O'$  and compare the sorted token vectors of concepts with each other. In case of multiple attribute values for a concept, the single similarities are aggregated to an overall similarity using the specified aggregation function `agg`. Our pre-processing allows for a very efficient overlap computation (Algorithm 2) similar to the Sort-Merge-Join used for efficient join computation in databases. Since all token sets are represented by sorted token vectors *stv*, we can do interleaved linear list scans to compute the overlap. We thus only perform  $|stv_1| + |stv_2|$  comparisons in the worst case with a fast integer-based token comparison. For instance, when comparing  $\{lim, imb\}$  with  $\{lim, imb, mbs\}$ , we compare  $[9,10]$  with  $[9,10,21]$  requiring merely the comparison of the two integer pairs 9-9 and 10-10.

GOMMA also supports the parallel execution of string matching for disjoint sets of concept pairs to utilize multiple processors or cores for improved execution time. In the evaluation (Sec. 4), we will also consider this performance option.

### 3 GPU-Based N-Gram Similarity Computation

A GPU-based implementation needs to overcome common GPU limitations, namely (1) lack of string data type, (2) only restricted data structures such as arrays, and (3) a priori allocation of a fixed and limited amount of memory. Since our algorithm operates on integer values, the first limitation is already

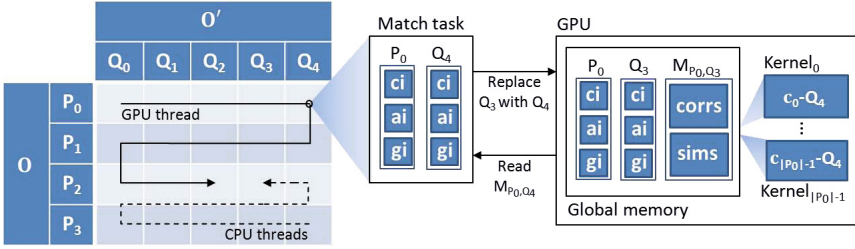


**Fig. 2.** GPU input and output data structures for running example and top- $k=2$

solved. For the second limitation we will use an index structure based on arrays. We will overcome the third limitation by partitioning large input ontologies, adapting memory-efficient data types, and determining only the best matches per concept to restrict the mapping size. We further use an execution scheme that minimizes expensive data transfers between main memory and GPU. In the following, we describe the utilized data structures and outline the n-gram similarity computation on GPUs.

**Input Data Structure:** In contrast to dynamically growing data structures (e.g., lists or maps) usable for CPU-based computations, GPU-based processing necessitates the preallocation of the required memory on the target device. Because the number of attributes per concept and the number of n-grams per attribute value varies, a mapping to fixed-length data structures is required. For this purpose, we adopt a multi-level index structure (illustrated in Fig. 2 for the running example) consisting of three arrays per input ontology: concept index ( $ci$ ), attribute index ( $ai$ ), and gram index ( $gi$ ). The arrays  $ci$  and  $ai$  represent the concepts and their attributes, respectively, while  $gi$  holds the sorted token vectors of the input concepts. For each concept, there is one entry in  $ci$  pointing to its first string attribute in  $ai$ . The number of attributes for concept  $j$  thus is  $ci[j+1]-ci[j]$ . Each  $ai$  entry represents a particular string and points to the first token of its value in  $gi$ . The last (dummy) entries of  $ci/ai$  are used to mark the end of each index. Using this structure, one can easily access the tokens of an attribute of a particular concept. For instance, to access the tokens of concept  $c_2 \in O$ , we first read  $ci[2]=3$  and  $ci[2+1]=5$  to find the lower (inclusive) and the upper (exclusive) bound of its attributes in  $ai$ . Hence, the concept has two attributes represented by  $ai[3]=8$  and  $ai[5-1]=10$ . The values at these positions can be used to access the sorted token vectors beginning at  $gi[8]$  and  $gi[10]$ , respectively. To save memory and transfer costs, we use short instead of integer data types to represent the tokens (2 instead of 4 bytes per token).

**Output Data Structure:** The memory for storing the match result must be reserved a priori as well. To limit the result size, we utilize the observation that it is sufficient to consider only the top- $k$  best correspondences (above threshold  $t$ ) for each concept without reducing match quality. This approach marks an



**Fig. 3.** Execution scheme for hybrid CPU/GPU-based n-gram similarity computation minimizing the data transfer between the host program and the GPU

upper bound of the required memory to allocate on the GPU. Our output data structure consists of two arrays `corrs` and `sims`. The former contains the ids of the (at most) top- $k$  matches per concept, the latter contains the corresponding similarities. For our running example, we would create two arrays of length 6 to store the best two matches for each concept of  $O$  (see bottom of Fig. 2). Again, the amount of memory and data transfer can be reduced by using the short data type (instead of float) to express the similarity values. In particular, we limit their precision to three decimal places which is sufficient for match processes, e.g., the similarity value 0.8 for  $c_2-c'_2$  is expressed by a short value of 800.

**N-Gram Execution on GPU:** Compared to CPUs, the architecture of GPU hardware exhibits a large number of simpler compute cores that execute the same instruction on multiple data partitions. In this study, we rely on the OpenCL framework for general purpose computation GPUs. OpenCL code is written in  $C$  as so-called compute kernels, whose submission is controlled by a host program executed on the CPU. The actual number of kernel instances running in parallel depends on the GPU's number of cores, its amount of memory, the kernel programs memory requirements, and the size of the input and output data. OpenCL assigns a global unique identifier to each kernel instance. This identifier is used to compute global memory offsets for loading and storing input data that a particular kernel is operating on.

In general, the input ontologies and the  $|O| \cdot k$  resulting correspondences exceed the available memory of the GPU. Thus, we up-front split both input ontologies into partitions  $P_i \subseteq O$  and  $Q_i \subseteq O'$ , analogously as in our previous work on parallel ontology matching [7]. We then iteratively ship pairs  $(P_i, Q_i)$  for comparison to the GPU. The GPU executes a kernel instance for each  $c \in P_i$  that compares  $c$  with all  $c' \in Q_i$  and determines its top- $k$  correspondences above  $t$ . The partial results are later unified by the host program. For this purpose, we utilize a job queue that supports the parallel n-gram similarity computation of different partition pairs on both the GPU as well as on the CPU. A dedicated thread takes match tasks from this queue and submits them to the GPU. In addition to this GPU thread, several CPU threads can access the job queue from the opposite end to independently perform matching on the CPU. We select jobs and ship partitions using the scheme displayed in Fig. 3. This scheme

ensures that after completion of a GPU job only a single partition needs to be transferred to the GPU. The other partition remains in the GPU’s memory and is reused for the next job. For instance, when the GPU finished the  $P_0$ - $Q_3$  job, it starts to execute  $P_0$ - $Q_4$  next. In this case, only the partition  $Q_3$  needs to be replaced by  $Q_4$  and  $P_0$  can be reused. Furthermore, it is beneficial to split the larger of the two input ontologies into partitions. If it even fits entirely into the device’s memory, only partitions from the smaller ontology need to be replaced.

## 4 Evaluation

We analyzed the execution time for computing the FMA-NCIT mapping which is part of the LargeBio match task in OAEI [14]. The task consists of three sub-tasks namely, **small** ( $3,720 \times 6,551$  concepts), **large** ( $28,885 \times 25,678$  concepts), and **whole** ( $79,042 \times 66,914$  concepts). To create mappings of high quality, we applied the GOMMA match workflow with  $\text{top-}k=1$  and  $n=3$  (Trigram) used in OAEI 2012 (for details and quality results see [6]). The experiments were carried out on an Intel i5-2500 machine (4x3.30GHz, 8GB memory). We further used the following mid-range GPU: Asus GTX660 with 960 CUDA cores/2GB memory.

The first experiment evaluates the execution times for the three sub-tasks utilizing either one CPU thread or the GPU. For CPU-based processing, we compare the proposed **SortInt** n-gram matching with two alternatives using nested-loop (**NLString**) and hash set look-ups (**HashString**) for computing the token set overlap. The results displayed in Fig. 4 (left) show that **SortInt**-CPU significantly outperforms both standard algorithms. For the **whole** task, it requires  $\approx 8$  min compared to about 26 min (104 min) for **HashString** (**NLString**), i.e., it improves runtime by up to a factor of 13. This shows that our pre-processing step pays off, i.e., converting strings into integer values and sorting are non-expensive ( $<1$  sec in all tasks) but valuable steps for an optimized overlap computation. The application of **SortInt** on the GTX660 GPU allows for a further significant improvement compared to the CPU implementation. The execution time for the **whole** task is reduced by another factor of 5 to merely 99 sec. Thus, transferring the data into the GPU pays off, i.e., the massively parallel hardware in the form of hundreds of CUDA cores substantially speeds up the computation.

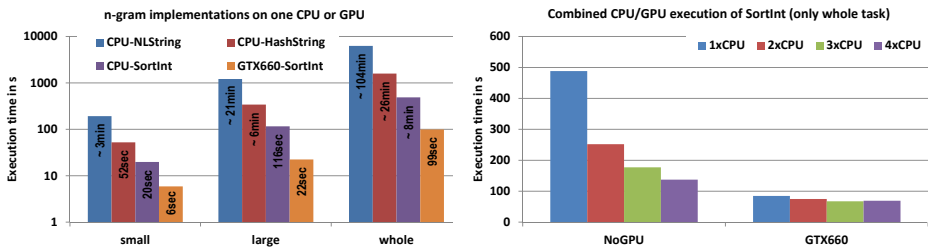


Fig. 4. Runtime of n-gram algorithms on CPU/GPU (left) and combined (right)

In a second experiment, we evaluate how application on multiple cores either without GPU (NoGPU) or in combination with GPU resources affects execution times. As shown in Fig. 4 (right), we observe that parallel CPU processing is very effective, e.g, when using four CPU threads, the execution time can be reduced to 137 sec (factor of 3.5) for the **whole** match task. The combined execution on CPU and GPU can further improve the execution time to about 67 sec for three and four CPU threads (factor of 2). The fourth CPU thread does not further improve the execution time due to the dedicated GPU thread for data transfer. Overall, one can see that even a moderately powered GPU can substantially reduce the execution time for string and thus for ontology matching.

## 5 Conclusion and Future Work

We studied how similarity functions like n-gram used for linguistic matching in GOMMA can be optimized by algorithmic tuning as well as by massively parallel processing on GPUs. The results indicate that intelligent pre-processing (e.g., integer conversion, sorting) of the input ontologies pays off substantially and speeds up ontology matching. The GPU-based execution of algorithms like n-gram matching requires some effort to overcome the GPU limitations but boosts performance even further. In the future we plan to investigate further GPU-based similarity computations and the impact of different kinds of GPU hardware.

## References

1. Ding, B., König, A.C.: Fast set intersection in memory. *PVLDB* 4(4) (2011)
2. Ehrig, M., Staab, S.: QOM – quick ontology mapping. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) *ISWC 2004*. LNCS, vol. 3298, pp. 683–697. Springer, Heidelberg (2004)
3. Euzenat, J., Shvaiko, P.: *Ontology matching*. Springer, New York (2007)
4. Forchhammer, B., et al.: Duplicate Detection on GPUs. In: *BTW* (2013)
5. Foundation Model of Anatomy, <http://fma.biostr.washington.edu/>
6. Gross, A., Hartung, M., Kirsten, T., Rahm, E.: GOMMA Results for OAEI 2012. In: *Proc. 7th Ontology Matching Workshop* (2012)
7. Gross, A., Hartung, M., Kirsten, T., Rahm, E.: On matching large life science ontologies in parallel. In: Lambrix, P., Kemp, G. (eds.) *DILS 2010*. LNCS, vol. 6254, pp. 35–49. Springer, Heidelberg (2010)
8. He, B., et al.: Relational joins on graphics processors. In: *Proc. SIGMOD* (2008)
9. Hu, W., Qu, Y., Cheng, G.: Matching large ontologies: A divide-and-conquer approach. *Data & Knowledge Engineering* 67(1) (2008)
10. Kirsten, T., Gross, A., Hartung, M., Rahm, E.: GOMMA: A Component-based Infrastructure for managing and analyzing Life Science Ontologies and their Evolution. *Journal of Biomedical Semantics* 2, 6 (2011)
11. NCI Thesaurus, <http://ncit.nci.nih.gov/>
12. Ngomo, A.-C.N., Kolb, L., Heino, N., Hartung, M., Auer, S., Rahm, E.: When to reach for the cloud: Using parallel hardware for link discovery. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) *ESWC 2013*. LNCS, vol. 7882, pp. 275–289. Springer, Heidelberg (2013)

13. Noy, N., et al.: BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research* 37(suppl. 2) (2009)
14. Ontology Alignment Evaluation Initiative, <http://oaei.ontologymatching.org/>
15. Owens, J., et al.: GPU computing. *Proceedings of the IEEE* 96(5) (2008)
16. Rahm, E.: Towards Large Scale Schema and Ontology Matching. In: *Schema Matching and Mapping*. Springer (2011)
17. Vouzis, P., Sahinidis, N.: GPU-BLAST: using graphics processors to accelerate protein sequence alignment. *Bioinformatics* 27(2) (2011)
18. Xiao, C., Wang, W., Lin, X., Yu, J.X., Wang, G.: Efficient similarity joins for near-duplicate detection. *ACM Trans. Database Syst.* 36(3) (2011)

# Semi-automatic Adaptation of Mappings between Life Science Ontologies

Anika Groß<sup>1</sup>, Julio Cesar Dos Reis<sup>2,3</sup>, Michael Hartung<sup>1</sup>,  
Cédric Pruski<sup>2</sup>, and Erhard Rahm<sup>1</sup>

<sup>1</sup> Department of Computer Science, University of Leipzig, Germany

<sup>2</sup> CR SANTEC, Public Research Centre Henri Tudor, Luxembourg

<sup>3</sup> LRI, University of Paris-Sud XI, France

{gross,hartung,rahm}@informatik.uni-leipzig.de,

{julio.dosreis,cedric.pruski}@tudor.lu

**Abstract.** The continuous evolution of life science ontologies requires the adaptation of their associated mappings. We propose two approaches for tackling this problem in a largely automatic way: (1) a *composition-based adaptation* relying on the principle of mapping composition and (2) a *diff-based adaptation* algorithm individually handling change operations to update the mapping. Both techniques reuse unaffected correspondences, and adapt only the affected mapping part. We experimentally assess and confirm the effectiveness of our approaches for evolving mappings between large life science ontologies.

**Keywords:** mapping adaptation, mapping migration, mapping evolution, ontology evolution, ontology mapping, ontology alignment.

## 1 Introduction

Ontologies and their applications have become increasingly important especially in the life sciences [1,2]. Typically there are many ontologies within a domain with overlapping information, *e.g.*, more than 30 anatomy-related ontologies in the OBO foundry [3]. Mappings between such related ontologies are useful for various data integration and enhanced analysis tasks. For instance, mappings are needed to merge several ontologies into an integrated ontology, *e.g.*, the multi-species anatomy ontology Uberon [4]. While manually curated mappings are especially valuable to interrelate the concepts of ontologies, it is often too time-consuming for large ontologies. Hence, semi-automatic matching approaches are increasingly needed for mapping creation [5,6,7].

The life sciences are a very dynamic field and new research results lead to a continuous evolution of ontologies so that new versions are periodically released [8]. Ontology changes include the addition, revision or deletion of concepts and relationships, and their frequency may substantially vary between ontologies or different parts of one ontology [9]. Ontology evolution can have an impact on different dependent artifacts such as ontology mappings [10,11], annotation mappings [12,13] and ontology-based queries [14,15]. As mappings may become



invalid and out-dated their adaptation is required. For example, a new version of an ontology in Biportal [16] or UMLS [17] may require the adaptation of the associated mappings, so that users and dependent applications can consume the most recent ones.

In this paper, we study different methods for a largely automatic adaptation of ontology mappings. In particular, we aim to avoid an expensive re-determination of the complete mapping and to reuse all stable parts from the old mapping. Migrating ontology mappings is not trivial for complex ontology changes such as the split of a concept into several new concepts. In this case an earlier correspondence with the unsplit concept may have to be changed to another or several new correspondences, and an expert user should be supported to select the correct result. Each type of ontology change may require different actions to update an ontology mapping. There is only little research so far on how to best perform the adaptation of mappings (see Sec. 2). Typically, previous approaches did not consider the impact of different ontology changes on mappings and also ignored new correspondences introduced by added concepts.

We therefore make the following contributions:

- We present a *composition-based* approach that uses ontology matching to create mappings between versions of an evolved ontology as well as the principle of mapping composition to create the adapted ontology mapping (Sec. 4).
- We propose a *diff-based* approach relied on a diff result consisting of the set of changes that led from the old to the new version of an ontology. The approach uses a library of change handlers to realize change-specific mapping adaptations (Sec. 5).
- We evaluate the approaches by adapting mappings between three large life science ontologies extracted from UMLS. Results reveal that we can adapt mappings largely automatically. We can also suggest specific mapping adaptations for certain types of ontology changes to simplify mapping curation (Sec. 6).

Additionally, we discuss related work in Sec. 2, present preliminaries on ontologies, mappings and the change model in Sec. 3, and conclude in Sec. 7.

## 2 Related Work

While a significant amount of research has already coped with the evolution of ontologies [18], the evolution of dependent mappings has received relatively little attention. In the context of schema evolution and model management [19,20], it has been proposed to evolve a previously determined mapping by composing it with a match mapping between the old and the new version of an updated schema or model. This composition approach has been explored in [21] for schema mappings and was shown to avoid the full re-calculation of existing mappings. We investigate and enhance the composition approach for adapting ontology mappings by not only reusing stable parts of the previous mapping, but by also extending the mapping, *e.g.*, for added ontology concepts.

Only few studies specifically investigated the maintenance and evolution of ontology mappings. In [22] the use of reasoners has been proposed for detecting and repairing invalid correspondences after ontology changes. Khattak *et al.* [23] propose to re-compute only those correspondences associated with changed ontology elements. Martins & Silva [24] propose that mapping evolution should behave similarly to strategies applied for ontology evolution. However, correspondences are only adapted when concepts are removed from the ontology. Kondylakis & Plexousakis [14] focus on the automatic detection of queries affected by ontology evolution. They assist developers to find and adapt invalid queries by suggesting sequences of changes affecting such queries.

In our previous work, we empirically analyzed which ontology changes lead to the addition or deletion of correspondences in an ontology mapping [11]. Dos Reis *et al.* [10] have proposed a framework for mapping evolution highlighting the role of different types of ontology changes for mapping adaptation, as well as the importance of considering different semantic types of correspondences in the adaptation process.

In contrast to prior studies, we not only aim at reusing stable parts of previous ontology mappings, but also extend the mappings for new ontology concepts. In addition to a composition-based method we propose a diff-based approach to individually handle different types of ontology changes and to solicit user feedback on adapted and newly determined correspondences. Unlike previous studies, we also evaluate the quality of the adapted mappings for large life science ontologies.

### 3 Preliminaries

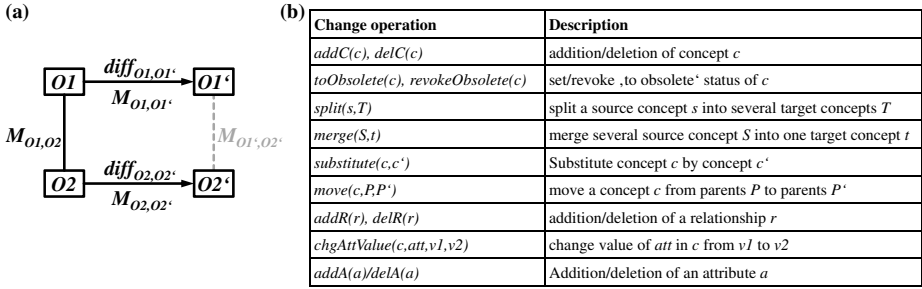
We first define the considered ontology and mapping model (Sec. 3.1) and then describe the general scenario we investigate in this paper (Sec. 3.2).

#### 3.1 Ontology Versions and Mappings

An ontology  $O = (C, R, A)$  consists of a set of concepts  $C$  interrelated by directed relationships  $R$ . Each concept  $c \in C$  is identified by an unambiguous accession number  $c_{acc}$ . Further attributes  $a \in A$  describe a concept in more detail, *e.g.*, labels, synonyms or definitions. A special attribute *obsolete* indicates whether a concept is outdated and should thus not be used anymore. A relationship  $r \in R$  interconnects two concepts and has a specific type, *e.g.*, 'is\_a' or 'part\_of'. An ontology version is a release of  $O$ , *i.e.*, a particular version is valid until a newer version becomes available. In the following, we denote two versions of an evolved ontology with  $O$  (old version) and  $O'$  (new version), respectively.

An ontology mapping  $M_{O_1, O_2}$  interconnects concepts of two different ontologies  $O_1/O_2$  by so-called correspondences:

$$M_{O_1, O_2} = \{(c_1, c_2, sim, semType, status) | c_1 \in O_1, c_2 \in O_2, sim \in [0, 1], \\ semType \in \{=, \leq, \geq, \approx\}, \\ status \in \{"handled", "toverify"\}\}$$



**Fig. 1.** (a) General scenario. (b) Considered change operations of COnto-Diff.

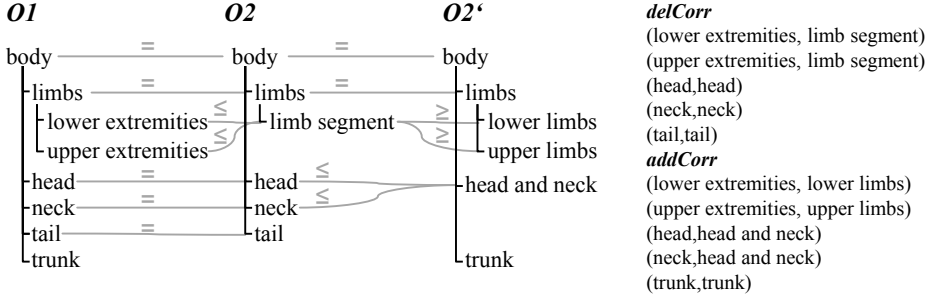
A correspondence  $(c_1, c_2, sim, semType, status)$  interrelates two concepts  $c_1 \in O1$  and  $c_2 \in O2$ . We use three further independent attributes to describe a correspondence in more detail. The  $sim$  value represents the similarity measure between  $c_1$  and  $c_2$ . The higher the value, the more related are both concepts. We assign a similarity of 1 to manually created correspondences. We further use a  $semType$  to differentiate the semantic connection type. For instance, concepts can be equivalent (*e.g.*, 'torso'='trunk'), one concept can be less or more general than the other (*e.g.*, 'thumb'≤'finger') or concepts can be somehow related ( $\approx$ ). A  $status$  signals the state of the correspondences during adaptation. In particular, a correspondence can be adapted (*handled*) or needs verification by an expert (*to verify*).

To create new mappings between ontologies we rely on semi-automatic match strategies because a purely manual mapping generation has become increasingly infeasible for large and complex ontologies [6,7]. For this purpose we use a successfully applied match strategy based on a concept's name and synonyms described in [25].

We also support the inversion of ontology mappings, *e.g.*, to get a mapping  $M_{O2,O1}$  out of  $M_{O1,O2}$ . To this end, we will use an inverse operator that inverts each correspondence as follows:  $(c_1, c_2, sim, semType, status) \mapsto (c_2, c_1, sim, newSemType, status)$ . In particular, the order of matching concepts is reversed, the similarity and the status values remain unchanged. The  $semType$  is adapted using the following rules:  $= \mapsto =$ ,  $\leq \mapsto \geq$ ,  $\geq \mapsto \leq$  and  $\approx \mapsto \approx$ .

### 3.2 General Scenario and Change Model

The general scenario investigated in this paper is depicted in Fig. 1a. There are two ontologies in their old ( $O1, O2$ ) and new versions ( $O1', O2'$ ). A mapping  $M_{O1,O2}$  interconnects the old versions of the two ontologies. The task investigated is to determine the new mapping  $M_{O1',O2'}$  which interrelates concepts of the new ontology versions  $O1'$  and  $O2'$ . For this purpose, we need further mappings between the ontology versions involved. In particular, there are two mappings  $M_{O1,O1'}$  and  $M_{O2,O2'}$  which interconnect concepts between the versions. These mappings provide information about how concepts in an old version



**Fig. 2.** Mapping evolution example

are related with concepts in the new version. We generate these mappings by matching, *i.e.*, we match  $O1$  with  $O1'$  and  $O2$  with  $O2'$ , respectively. The proposed composition-based approach (Sec. 4) uses the mappings  $M_{O1,O2}$ ,  $M_{O1,O1'}$  and  $M_{O2,O2'}$  to create the adapted mapping  $M_{O1',O2'}$  based on composition.

We further use so-called evolution mappings ( $diff_{O1,O1'}$  and  $diff_{O2,O2'}$ ) between the old and new ontology versions. These mappings integrate all changes that occurred during evolution from  $O1$  to  $O1'$  and  $O2$  to  $O2'$ , respectively. An evolution mapping can be created using a Diff tool such as PromptDiff [26] or COnto-Diff [27] and contains different types of changes (Fig. 1b lists changes of COnto-Diff). For instance, there are concept changes such as add, delete, merge and split, or changes of attribute values. The proposed diff-based approach (Sec. 5) uses the diff evolution mappings  $diff_{O1,O1'}$  and  $diff_{O2,O2'}$  to create the adapted mapping  $M_{O1',O2'}$ .

## 4 Composition-Based Adaptation

This section presents the composition-based approach for mapping adaptation. Its strength is the reuse of the previous, already validated ontology mapping to avoid an expensive re-computation of confirmed correspondences. Given that changes are typically limited to a small subset of ontologies, this promises that the largest part of the new mapping is easily determined. For illustration purpose, we use a running example shown in Fig. 2 with an evolution of an anatomy ontology ( $O2 \mapsto O2'$ ). The ontology changes require an adaptation of the mapping  $M_{O1,O2}$ , in particular to delete the previous correspondence ( $delCorr$ ) and to add the new correspondence ( $addCorr$ ) shown on the right side. Our composition-based approach achieves the adaptation by composing the previous ontology mapping  $M_{O1,O2}$  with the mapping  $M_{O2,O2'}$ , as well as by checking whether added concepts lead to new correspondences.

The composition of two mappings  $M_{A,B}$  and  $M_{B,C}$  generates a mapping  $M_{A,C}$  between  $A$  and  $C$ . With mappings as introduced in Sec. 3.1, we define:

$$M_{A,C} = \text{compose}(M_{A,B}, M_{B,C}) = M_{A,B} \circ M_{B,C} = \{(c_1, c_2, \text{aggSim}(\text{sim}_1, \text{sim}_2), \text{getNewType}(\text{semType}_1, \text{semType}_2)),$$

$$\begin{aligned} & \text{getNewStatus}(\text{semType}_1, \text{semType}_2)) | \\ c_1 \in A, c_2 \in C, b \in B : & \exists (c_1, b, \text{sim}_1, \text{semType}_1, \text{status}_1) \in M_{A,B} \wedge \\ & \exists (b, c_2, \text{sim}_2, \text{semType}_2, \text{status}_2) \in M_{B,C} \} \end{aligned}$$

The generation of a correspondence  $(c_1, c_2)$  in  $M_{A,C}$  requires the existence of two correspondences  $(c_1, b)$  and  $(b, c_2)$  connecting to the same concept  $b \in B$ . The attribute values of the new correspondence are derived from the values of the two 'connecting' correspondences. First, the new similarity is aggregated from the similarities  $\text{sim}_1$  and  $\text{sim}_2$  by computing, *e.g.*, their average or maximum (**aggSim**). Second, the new semantic type is derived from  $\text{semType}_1$  and  $\text{semType}_2$  (**getNewType**) based on the rule set presented in Fig. 4a. For example, the combination of '=' and '<=' would lead to the new semantic type '<='. Third, the new correspondence is assigned the new status (**getNewStatus**, see Sec.5.2).

**CompAdapt** (Algorithm 1) shows how we perform composition-based mapping adaption for the general case when both ontologies evolve ( $O1 \mapsto O1'$ ,  $O2 \mapsto O2'$ ). The algorithm uses as input the previous ontology mapping  $M_{O1,O2}$  as well as the two mappings  $M_{O1,O1'}$  and  $M_{O2,O2'}$ .

---

**Algorithm 1.**  $\text{CompAdapt}(M_{O1,O2}, M_{O1,O1'}, M_{O2,O2'})$

---

```

1  $M_{O1',O1} \leftarrow \text{inverse}(M_{O1,O1'})$ ;
2  $M_{O1',O2} \leftarrow \text{compose}(M_{O1',O1}, M_{O1,O2})$ ;
3  $M_{O1',O2'} \leftarrow \text{compose}(M_{O1',O2}, M_{O2,O2'})$ ;
4 return  $M_{O1',O2'}$ ;

```

---

We first generate the inverse mapping  $M_{O1',O1}$  (line 1) and compose it with  $M_{O1,O2}$  to create an intermediate mapping between  $O1'$  and  $O2$  (line 2). We then transitively compose the intermediate mapping with  $M_{O2,O2'}$  to produce the adapted mapping  $M_{O1',O2'}$  between  $O1'$  and  $O2'$  (line 3). When exclusively one of the input ontologies evolve, we only need one of the two compositions. We perform the first two steps if  $O1$  evolves to  $O1'$ , or only perform  $\text{compose}(M_{O1,O2}, M_{O2,O2'})$  if  $O2$  evolves to  $O2'$ . For the running example (Fig. 2), we would create eight correspondences including retained correspondences such as ('limbs', 'limbs'). Unfortunately, the composition also creates the false correspondences (('lower extremities', 'upper limbs'), ('upper extremities', 'lower limbs')) since the concept 'limb segment' in the intermediate ontology is connected to several concepts in the ontologies to be composed. We will later see how our alternate solution (Sec. 5) can cope with such situations.

Composition alone is also unable to determine new correspondences due to added concepts in the ontologies, *e.g.*, 'trunk' in  $O2'$ . To address this shortcoming we apply an additional match step as shown in the **CompAdaptMatch** algorithm:

---

**Algorithm 2.**  $\text{CompAdaptMatch}(M_{O1,O2}, M_{O1,O1'}, M_{O2,O2'}, O1, O1', O2, O2')$

---

```

1  $M_{O1',O2'} \leftarrow \text{CompAdapt}(M_{O1,O2}, M_{O1,O1'}, M_{O2,O2'})$ ;
2  $\text{Add}_{O1} \leftarrow O1' \setminus O1$ ;
3  $\text{Add}_{O2} \leftarrow O2' \setminus O2$ ;
4  $M_{O1',O2'} \leftarrow M_{O1',O2'} \cup \text{match}(\text{Add}_{O1}, O2') \cup \text{match}(O1', \text{Add}_{O2})$ ;
5 return  $M_{O1',O2'}$ ;

```

---

After adapting the mapping using composition (line 1) we identify the added concepts ( $Add_{O_1}, Add_{O_2}$ ) in both ontologies (lines 2–3). We match the added concepts with the other ontology to find new correspondences (line 4) and include them in the adapted mapping. We can simplify the algorithm when exclusively one of the ontologies has changed by merely matching added concepts of the changed ontology with the unchanged ontology. In the running example, we would determine 'trunk' as an added concept in  $O_2'$  and matching would result in the additional correct correspondence ('trunk', 'trunk') in the adapted mapping.

## 5 Diff-Based Adaptation

The Diff-based adaptation of ontology mappings considers the individual ontology changes, and so-called change handlers to adapt the ontology mapping. This modular approach is highly flexible and can accommodate different types of changes as well as distinct automatic or interactive approaches for mapping adaptation. For example, a concept deletion would lead to the deletion of all affected correspondences with the composition-based approach, while a change handler could try to keep a correspondence with a neighbor of the deleted concept. Furthermore, change handlers might request expert verification for proposed mapping changes.

We first explain Diff-based mapping adaptation for the frequent case when only one of two ontologies changes (Sec. 5.1). We then explain the different change handlers and their approaches for mapping adaptation (Sec. 5.2). Finally, we discuss Diff-based adaptation for the general case with two evolving ontologies (Sec. 5.3). Although the proposed approach is applicable for different diff techniques to determine ontology changes, we assume the use of our algorithm COnTo-Diff [27] for concreteness. COnTo-Diff is suited to identify a diff evolution mapping for two successive versions of an ontology containing typical change operations such as *merge*, *substitute*, *split*, *addC* or *delC* (see Fig. 1b).

### 5.1 Adaptation Algorithm for One Evolving Ontology

The input data of the algorithm DiffAdapt (Algorithm 3) are the ontology mapping to be adapted ( $M_{O_1, O_2}$ ), the two versions of the domain ontology  $O_1, O_1'$ , a diff between them ( $diff_{O_1, O_1'}$ ) as well as the current version of the range ontology  $O_2$ . We assume that the change handlers are listed in the order in which they should be applied for mapping adaptation ( $CH$ ). This ordering is feasible since COnTo-Diff ensures that a concept is the subject of at most one of the considered change operations.

---

#### Algorithm 3. DiffAdapt( $M_{O_1, O_2}, diff_{O_1, O_1'}, O_1, O_1', O_2, CH$ )

---

```

1  $M_{infl} \leftarrow \text{getInfluencedCorrs}(M_{O_1, O_2}, diff_{O_1, O_1'}, CH)$ ;
2  $M_{O_1', O_2} \leftarrow M_{O_1, O_2} \setminus M_{infl}$ ; //reuse unaffected mapping part
3 foreach  $ch \in CH$  do
4    $diffPart \leftarrow diff.filter(ch.getHandledOperations());$ 
5    $ch.handleChg(M_{infl}, diffPart_{O_1, O_1'}, O_1, O_1', O_2)$ ;
6  $M_{O_1', O_2} \leftarrow M_{O_1', O_2} \cup M_{infl}$ ;
7 return  $M_{O_1', O_2}$ ;

```

---

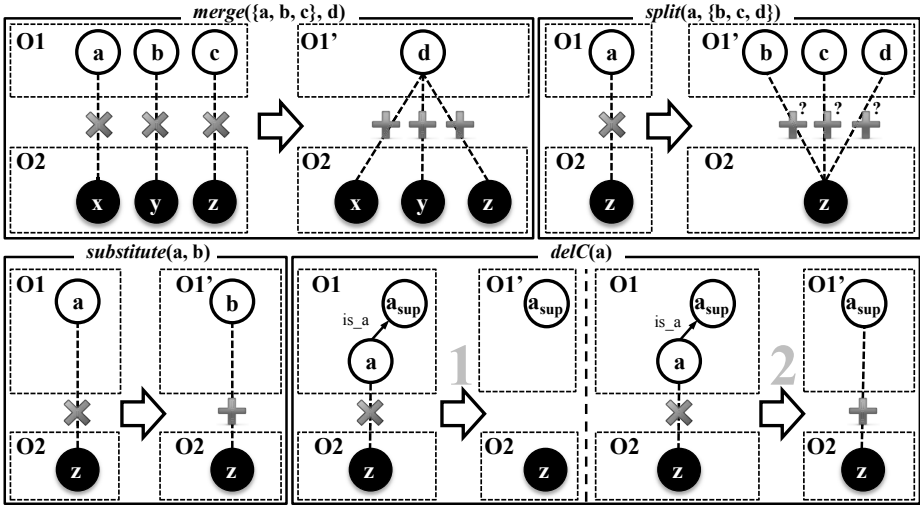


Fig. 3. Change handlers

We first identify all correspondences that are influenced by changes from the input diff. Therefore, we check if the domain concept of each correspondence was subject to a change operation listed in CH. All influenced correspondences in  $M_{in\_fl}$  are initially set to status *to verify*, since they might require user verification. By contrast, we reuse unaffected correspondences (status *handled*) by adding them directly to the new mapping  $M_{O1',O2}$  (line 2). For instance, in the running example (Fig. 2), 'limbs' and 'body' remain unchanged in  $O2$  so that we keep the correspondences ('limbs', 'limbs') and ('body', 'body'). The influenced mapping part  $M_{in\_fl}$  is then handled by the specified list of individual *Change Handlers* (lines 3-5). The mapping  $M_{in\_fl}$  is iteratively adapted, *i.e.*, each change handler removes outdated correspondences from and adds new correspondences to  $M_{in\_fl}$ . Depending on the used method in the change handler, the status of new correspondences is either set to *handled* or *to verify*. Finally, we take the union of the reused correspondences in  $M_{O1',O2}$  and the adapted mapping part  $M_{in\_fl}$  and then return the resulting mapping (lines 6-7).

## 5.2 Change Handlers

We provide a handler for each type of ontology change to implement appropriate approaches for mapping adaptation. These handlers can easily be adapted and extended to adjust mapping adaptation, request users' feedback in certain cases or deal with new types of ontology changes. Fig. 3 illustrates main adaptation choices for some major change operations namely *merge*, *substitute*, *split* and *delC*. It shows how correspondences from  $M_{O1,O2}$  are adapted according to the evolution from  $O1$  to  $O1'$ . In the following, we present the change handlers

in the order in which they are applied in the algorithm DiffAdapt:  $CH_{merge}$ ,  $CH_{substitute}$ ,  $CH_{split}$ ,  $CH_{delC}$ ,  $CH_{toObsolete}$ ,  $CH_{addC}$  and  $CH_{revokeObsolete}$ .

In the *merge* operation, two or more source concepts from  $O1$  are merged into one target concept in  $O1'$ . The merge handler migrates all correspondences once associated with any of the  $O1$  concepts to the target concept in  $O1'$ . Thus, each correspondence from  $M_{O1,O2}$  associated with concepts to be merged are removed and new correspondences to the target concept are added. In the running example (Fig. 2) 'head' and 'neck' concepts are merged as 'head and neck'. All correspondences once related to 'head' or 'neck' are assigned to the new concept 'head and neck'. Algorithm 4 details the sketched approach of the merge handler. It checks for each correspondence *corr* (line 1) and merge operation *merge* (line 2) if the domain concept of *corr* is equal to one of the source concepts in *merge* (lines 5-6). If so, the affected correspondence is adapted.

---

**Algorithm 4.** MergeHandler( $M, Merge, O1, O1'$ )

---

```

1  foreach corr ∈ M do
2      foreach merge ∈ Merge do
3          S ← merge.getSourceIDs();
4          t ← merge.getTargetID();
5          foreach s ∈ S do
6              if s = corr.getDomainID() then
7                  newType ← getNewType(corr.getType(), ≤);
8                  newStatus ← getNewStatus(corr.getType(), ≤);
9                  newCorr ← createCorr(t, corr.getRangeID(),
10                     corr.getSim(), newType, newStatus);
11                 M.remove(corr).add(newCorr);

```

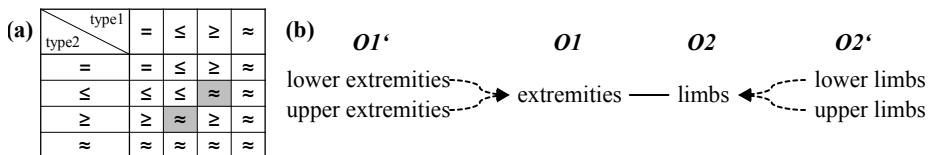
---

The merge handler supports an adaptation of the semantic type of added correspondences. For example, for  $merge(\{a, b, c\}, d)$  it usually holds that concepts  $a, b, c$  are less general ( $\leq$ ) than  $d$ . Hence, we combine  $\leq$  with the semantic type of the old correspondence ( $=, \leq, \geq, \approx$ ) to derive the new semantic correspondence type.

Such an adaptation of the semantic correspondence type is needed for different types of changes and was also applied for mapping composition. To combine semantic types of correspondences (operation *getNewType*) and to determine the new correspondence status (operation *getNewStatus*) we currently use a set of combination rules as shown in Fig. 4a. The basic idea is that the semantic type with lower binding strength imposes the new semantic type. Following the definition of *semantic relation* in [28],  $=$  has a higher binding strength than  $\leq$  and  $\geq$  which in turn are stronger than  $\approx$ .  $\leq$  and  $\geq$  are of equal binding strength such that the new semantic type of their combination can not be determined by rules (gray fields). The status *to verify* is set to  $\approx$  since a user necessarily needs to check this correspondence and its semantic type. For all other combinations as shown in Fig. 4a, the status of the correspondence is *handled*.

For the  $substitute(a, b)$  change operation, the applied strategy is similar to the one used for *merge*. In this case, the concept  $a \in O1$  is substituted by the target concept  $b \in O1'$ . Since  $a$  is involved in a correspondence with  $z$  in  $O2$ , the correspondence between  $a$  and  $z$  is removed and the new correspondence from  $b$





**Fig. 4.** (a) Combining semantic types (`getNewType`) and determine the new correspondence status (`getNewStatus`). (b) Example of conflicting changes for two evolving ontologies.

to  $z$  is added. We can assume  $a = b$  as semantic type for substitute, and combine this with the old semantic type of the correspondence to derive the new one.

The adaptation of correspondences affected by split change operations is more complex. For example,  $split(a, \{b, c, d\})$  caused a single source concept  $a \in O1$  to be split into several target concepts  $b, c, d \in O1'$ . In the mapping adaptation, we first remove all correspondences associated with the split source concept  $a$ . We consider two strategies for adding new correspondences. First, one can add all possible combinations of correspondences between the split target concepts  $b, c, d$  and the unmodified range concept  $z$  in  $O2$  ("take all"). Second, we can restrict the output result to the best correspondence(s), *i.e.*, the one(s) with the highest similarity based on a local match between  $b, c, d$  and  $z$  ("take best").

Also for split, new adapted correspondences obtain an individual new semantic type based on the rules in Fig. 4a and assuming that  $d \geq a, b, c$  holds for split. All correspondences get status *to verify* since these are only recommendations and an expert needs to decide about their validity. In the running example (Fig. 2) 'limb segment' was split into 'lower limbs' and 'upper limbs'. Using the "take all" strategy, we would present all four possible combinations between 'lower extremities', 'upper extremities' and 'lower limbs', 'upper limbs' to the user. Using the "take best" strategy, we can correctly identify the most adequate correspondences 'lower limbs' with 'lower extremities', and 'upper limbs' with 'upper extremities'.

For deletion of concepts ( $delC(a)$ ) we also consider two strategies. First, all correspondences referencing deleted concepts in  $O1'$  are removed (see Fig. 3) (strategy "del corr"). This is the case for 'tail' in the running example. Second, correspondences can be transferred to their parent concept, if possible ("keep corr"). Thus, correspondences related to the deleted concept  $a$  are removed, but new "more general" correspondences are created. In particular, the domain of the new correspondence is the first super concept ( $a_{sup}$ ) of  $a$ . In case of multiple inheritance, the correspondence can be transferred to all parents. The status is set to *to verify* since a user has to check the adapted correspondences. The new semantic type is derived by following the  $\leq$  parent relationship in  $O1$  combined with the semantic type of the old deleted correspondence. For *to Obsolete* changes we apply the same handler.

For all concept additions and *revokeObsolete* operations in  $O1'$  we apply an automatic matching step with the whole range ontology  $O2$ . The status of the new recommended correspondences is set to *to verify*. One can either apply a

very restrictive selection of correspondences to show only the best matches to experts, and avoid many false positives, or to be less restrictive in order to get a perfect recall and let the selection up to the user. In the running example,  $diff_{O_2, O_2'}$  contains an addition of the concept 'trunk' which is matched to  $O_1$  such that ('trunk', 'trunk') is correctly identified by selecting only the top result.

### 5.3 Adaptation Algorithm for Two Evolving Ontologies

In case where both ontologies change (domain and range of the correspondences), we can adapt the mapping by applying the DiffAdapt (Algorithm 3) twice as follows:

---

**Algorithm 5.** DiffAdaptBoth( $M_{O_1, O_2}, diff_{O_1, O_1'}, diff_{O_2, O_2'}, O_1, O_1', O_2, O_2', CH$ )

---

```

1  $M_{O_1', O_2} \leftarrow \text{DiffAdapt}(M_{O_1, O_2}, diff_{O_1, O_1'}, O_1, O_1', O_2, CH)$ ;
2  $M_{O_2, O_1'} \leftarrow \text{inverse}(M_{O_1', O_2})$ ;
3  $M_{O_2', O_1'} \leftarrow \text{DiffAdapt}(M_{O_2, O_1'}, diff_{O_2, O_2'}, O_2, O_2', O_1', CH)$ ;
4 return  $\text{inverse}(M_{O_2', O_1'})$ ;

```

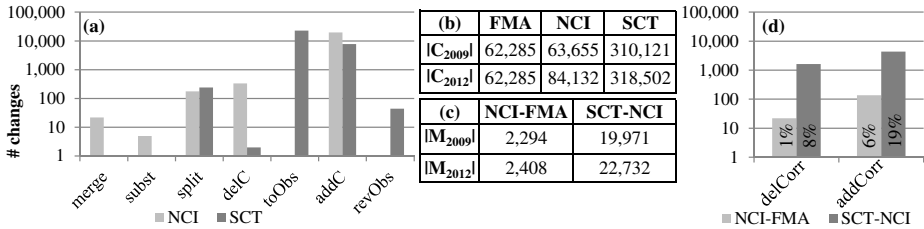
---

The input of algorithm DiffAdaptBoth (Algorithm 5) is similar as for DiffAdapt but requires two versions for both input ontologies  $O_1, O_1', O_2, O_2'$ , as well as two diff mappings  $diff_{O_1, O_1'}/diff_{O_2, O_2'}$ . First, we adapt the given ontology mapping with respect to changes in the domain ontology to get  $M_{O_1', O_2}$ . To adapt the mapping regarding changes in the range ontology we call DiffAdapt with the inverse mapping  $M_{O_2, O_1'}$  and the range diff  $diff_{O_2, O_2'}$  (line 3). Finally, we invert the mapping again and return it (line 4).

When both ontologies change, some correspondences might be affected by changes of the domain and range concept at the same time. For instance, if both concepts of a correspondence are split into several concepts, we can produce wrong results by independently handling these changes one after the other. A possible problem scenario is shown in Fig. 4b. Applying the "take all" strategy twice would create too many correspondences, namely the local cross-product. By contrast, "take best" might lead to a wrong selection of ('lower extremities', 'limbs') in the first step, such that we can only find ('lower extremities', 'lower limbs') after the adaptation concerning the range ontology. To deal with such situations when both ontologies have evolved, we propose to handle these conflicting changes together in an extra step. We can first identify correspondences involved in conflicts and modify the input mapping before we run DiffAdaptBoth. In particular, we recommend to check conflicting change combinations as *split-split*, *merge-split* and *substitute-split* where it is helpful to do the migration on both sides in one step.

## 6 Evaluation

To evaluate the proposed approaches for mapping adaptation, we use three large life science ontologies: SNOMED-CT (SCT), NCI Thesaurus (NCI) and FMA. We use the integrated ontology UMLS to extract two mappings NCI-FMA and



**Fig. 5.** (a) Ontology changes (b) Ontology size (c) Mapping size (d) Mapping changes

SCT-NCI in two versions for 2009 and 2012 (see [29] for extraction details). We adapt the mapping versions from 2009 with the proposed algorithms, and use the 2012 versions as reference mappings for evaluating the quality of the mappings adapted. It is important to notice that such reference mappings can be considered as a 'silver standard', *i.e.*, these mappings are not complete, and curators manually correct them by modifying also correspondences associated with concepts that did not underlie changes. In this evaluation we eliminate such correspondences from the mappings since they do not change due to ontology modifications and can thus not be detected. To assess the quality of the adapted mappings with respect to the 2012 reference mappings, we calculate the standard metrics of Precision, Recall and F-Measure.

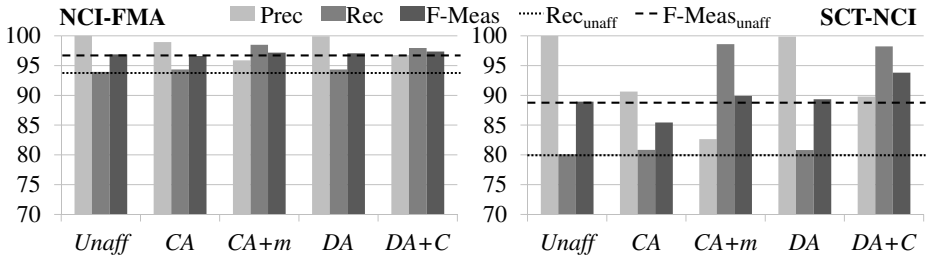
In the following we first analyze the used data sets (Sec. 6.1) and then evaluate the quality of the proposed mapping adaptation approaches (Sec. 6.2).

## 6.1 Ontology and Mapping Analysis

Fig. 5 gives an overview of changes in the considered ontology versions (a) and mapping versions (d) as well as of their sizes (b,c). From 2009 to 2012, FMA remains completely stable while NCI and SCT have been revised considerably. Besides some merge operations (22 for NCI) there was a notable number of  $\sim 180$  (240) concept splits for NCI (SCT). In SCT an enormous amount of  $>22.000$  concepts has been set to obsolete while NCI has been extended by  $\sim 20.000$  concepts during 2009 and 2012. The 2009 mapping version of NCI-FMA is relatively small ( $\sim 2300$ ) compared to SCT-NCI ( $\sim 20400$ ) (Fig. 5c). During the considered time interval of three years, the NCI-FMA mapping grew by  $\sim 5\%$  and SCT-NCI by even  $14\%$ . The SCT-NCI mapping has been affected by more changes, namely  $8\%$  of the correspondences have been deleted from the old and  $19\%$  were added to the new mapping version. Thus, NCI-FMA has a higher rate of unchanged correspondences and might be easier to adapt than SCT-NCI.

## 6.2 Mapping Adaptation Results

Fig. 6 shows the quality of the mapping adaptation results for NCI-FMA (left) and SCT-NCI (right). To have a basic reference for analyzing how much each



**Fig. 6.** Results on the Quality of Mapping Adaptation

adaptation approach contributes, we mark the impact of unaffected (stable) correspondences in the adapted mapping (*Unaff*). The dotted and dashed lines highlight the recall ( $Rec_{unaff}$ ) and F-Measure ( $F-Meas_{unaff}$ ) of *Unaff*. We compare results with the composition-based adaptation (CA) and its match extension (CA+m). Moreover, we apply the diff-based adaptation (DA) using the major handlers  $CH_{merge}$ ,  $CH_{substitute}$ ,  $CH_{split}$  ("take best"),  $CH_{delC}$  and  $CH_{toObsolete}$  ("del corr"), and as an extension (DA+C) the  $CH_{addC}$  and  $CH_{revokeObsolete}$  handlers. Note that our approach is flexible and can be easily extended to handle also attribute and structural changes. In the evaluation scenario, this showed to have a negative impact on the quality of adapted mappings, such that we omit it in this study. We consider this an issue for future investigations.

For both cases analyzed, the basic quality of *Unaff* is already very high, since 94% (80%) of the NCI-FMA (SCT-NCI) mappings were unaffected and could be reused. For the adaptation of the relatively stable NCI-FMA mapping all considered approaches perform similarly well and achieve a very high F-Measure. SCT-NCI is a more challenging mapping adaptation scenario and helps to better differentiate the relative effectiveness of the proposed approaches. Compared to *Unaff*, CA is less precise and increases the recall only marginally. This is caused by the fact that the applied compose approach takes all possible combinations of existing correspondences, and no further selection takes place. An additional match of new concepts (CA+m) significantly increases the recall by 18.6% for SCT-NCI and slightly improve F-Measure compared to *Unaff* (despite a reduced precision for automatically generated match correspondences).

For SCT-NCI, the diff-based approaches clearly outperform the composition-based approaches. They not only reuse unaffected correspondences but can further improve recall with relatively high precision due to the individual change handling. DA+C performs best overall since it utilizes additional change handlers. In particular, it can find additional match correspondences for added concepts leading to a significant increase in recall and F-Measure. While this is similar to the high recall of CA+m, the precision and thus F-Measure remains higher for DA+C ( $\sim 94\%$  instead of  $\sim 90\%$ ). The recall could even be further increased by using a lower match threshold than the applied 1.0, and let experts select the correct correspondences out of the recommended matches in DA+C.

Based on these results, we recommend that ontology mappings might be adapted in a semi-automatic manner as follows: (1) first, determine a consistent adapted mapping using the DA approach; (2) apply further strategies such as DA+C that provide recommendations of new correspondences; (3) apply expert knowledge based on the adaptation results to complete the mapping and validate those correspondences with *to verify* status.

## 7 Conclusion

Ontology evolution can potentially invalidate previously created mappings. We proposed a *composition- and a diff-based* approach for adapting ontology mappings as a consequence of ontology evolution. Both approaches can reuse unaffected correspondences from existing mappings and adapt only the changed parts in a (semi-)automatic way. The composition-based approach is conceptually simpler but can be already sufficient for ontologies that change only slightly. The diff-based approach is more powerful by supporting different change-specific approaches for mapping adaptation and by enabling experts to verify proposed correspondences. The conducted evaluation for large life science ontologies confirmed the high effectiveness of the proposed approaches. Both of them benefit from matching new concepts to produce a more complete mapping.

For future work, we plan to realize a more refined adaptation of semantic mappings. The techniques presented already support the migration of semantic mappings, but this has to be investigated in more detail and evaluated for real-world semantic mappings. Additionally, in further evaluation expert users should analyze the quality of mappings for the different adaptation strategies.

**Acknowledgment.** This work is funded by the German Research Foundation (DFG) (grant RA 497/18-1, "Evolution of Ontologies and Mappings"), by the National Research Fund (FNR) of Luxembourg (grant C10/IS/786147), by the European Social Fund and the Free State of Saxony.

## References

1. Bodenreider, O., Stevens, R.: Bio-ontologies: current trends and future directions. Briefings in Bioinformatics 7(3) (2006)
2. Lambrix, P., Tan, H., Jakoniene, V., Strömbäck, L.: Biological Ontologies. In: Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences (2007)
3. Smith, B., et al.: The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. Nature Biotechnology 25(11) (2007)
4. Mungall, C., et al.: Uberon, an integrative multi-species anatomy ontology. Genome Biol. 13(1) (2012)
5. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. The VLDB Journal 10(4) (2001)
6. Euzenat, J., Shvaiko, P.: Ontology matching. Springer, New York (2007)
7. Rahm, E.: Towards Large Scale Schema and Ontology Matching. In: Schema Matching and Mapping. Springer (2011)
8. 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)

9. Malone, J., Stevens, R.: Measuring the level of activity in community built bio-ontologies. *J. Biomed. Inform.* 46(1) (2013)
10. Dos Reis, J., Pruski, C., Da Silveira, M., Reynaud, C.: Analyzing and Supporting the Mapping Maintenance Problem in Biomedical Knowledge Organization Systems. In: *Proc. SIMI Workshop at ESWC* (2012)
11. Groß, A., Hartung, M., Thor, A., Rahm, E.: How do computed ontology mappings evolve?-A case study for life science ontologies. In: *Joint Workshop on Knowledge Evolution and Ontology Dynamics* (2012)
12. Groß, A., Hartung, M., Kirsten, T., Rahm, E.: Estimating the quality of ontology-based annotations by considering evolutionary changes. In: Paton, N.W., Missier, P., Hedeler, C. (eds.) *DILS 2009. LNCS*, vol. 5647, pp. 71–87. Springer, Heidelberg (2009)
13. Groß, A., Hartung, M., Prüfer, K., Kelso, J., Rahm, E.: Impact of Ontology Evolution on Functional Analyses. *Bioinformatics* 28(20) (2012)
14. Kondylakis, H., Plexousakis, D.: Ontology Evolution: Assisting Query Migration. In: Atzeni, P., Cheung, D., Ram, S. (eds.) *ER 2012. LNCS*, vol. 7532, pp. 331–344. Springer, Heidelberg (2012)
15. Liang, Y., Alani, H., Shadbolt, N.: Changing ontology breaks queries. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) *ISWC 2006. LNCS*, vol. 4273, pp. 982–985. Springer, Heidelberg (2006)
16. Noy, N., et al.: BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Res.* 37(suppl. 2) (2009)
17. Bodenreider, O.: The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic Acids Research* 32(suppl. 1) (2004)
18. Hartung, M., Terwilliger, J.F., Rahm, E.: Recent Advances in Schema and Ontology Evolution. In: *Schema Matching and Mapping*. Springer (2011)
19. Velegrakis, Y., Miller, J., Popa, L.: Mapping Adaptation under Evolving Schemas. In: *Proc. VLDB* (2003)
20. Bernstein, P., Melnik, S.: Model management 2.0: manipulating richer mappings. In: *Proc. SIGMOD* (2007)
21. Yu, C., Popa, L.: Semantic Adaptation of Schema Mappings when Schemas Evolve. In: *Proc. VLDB* (2005)
22. Meilicke, C., Stuckenschmidt, H., Tamilin, A.: Reasoning Support for Mapping Revision. *Journal of Logic and Computation* 19(5) (2008)
23. Khattak, A., Pervez, Z., Latif, K., Lee, S.: Time efficient reconciliation of mappings in dynamic web ontologies. *Knowl.-Based Syst.* 35 (2012)
24. Martins, H., Silva, N.: A User-Driven and a Semantic-Based Ontology Mapping Evolution Approach. In: *Proc. Intl. Conf. on Enterprise Inform. Systems* (2009)
25. Groß, A., Hartung, M., Kirsten, T., Rahm, E.: GOMMA results for OAEI 2012. In: *Proc. OM Workshop at ISWC*, vol. 11 (2012)
26. Noy, N.F., Musen, M.A.: Promptdiff: A fixed-point algorithm for comparing ontology versions. In: *Proc. of Nat. Conf. on Artificial Intelligence* (2002)
27. Hartung, M., Groß, A., Rahm, E.: COnto-Diff: generation of complex evolution mappings for life science ontologies. *J. Biomed. Inform.* 46(1) (2013)
28. Giunchiglia, F., Shvaiko, P., Yatskevich, M.: S-Match: an algorithm and an implementation of semantic matching. In: Bussler, C.J., Davies, J., Fensel, D., Studer, R. (eds.) *ESWS 2004. LNCS*, vol. 3053, pp. 61–75. Springer, Heidelberg (2004)
29. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga, R.: Logic-based assessment of the compatibility of UMLS ontology sources. *J. Biomed. Sem.* 2 (2011)

# Next Generation Cancer Data Discovery, Access, and Integration Using Prizms and Nanopublications

Jamie P. McCusker<sup>1,3</sup>, Timothy Lebo<sup>2</sup>,  
Michael Krauthammer<sup>3</sup>, and Deborah L. McGuinness<sup>1,2</sup>

<sup>1</sup> Department of Computer Science

<sup>2</sup> Department of Cognitive Science, Rensselaer Polytechnic Institute,  
110 8th Street Troy, NY 12180, USA

<http://tw.rpi.edu>

<sup>3</sup> Department of Pathology, Yale School of Medicine, 300 George St., New Haven,  
CT, 06510, USA

<http://krauthammerlab.med.yale.edu>

{mccusj,lebot}@rpi.edu, dlm@cs.rpi.edu, michael.krauthammer@yale.edu

**Abstract.** To encourage data sharing in the life sciences, supporting tools need to minimize effort and maximize incentives. We have created infrastructure that makes it easy to create portals that supports dataset sharing and simplified publishing of the datasets as high quality linked data. We report here on our infrastructure and its use in the creation of a melanoma dataset portal. This portal is based on the Comprehensive Knowledge Archive Network (CKAN) and Prizms, an infrastructure to acquire, integrate, and publish data using Linked Data principles. In addition, we introduce an extension to CKAN that makes it easy for others to cite datasets from within both publications and subsequently-derived datasets using the emerging nanopublication and World Wide Web Consortium provenance standards.

## 1 Introduction

Peer-reviewed publications remain the principal means for exchanging cancer research information, despite the critical need for other researchers to access supporting data so that they may progress their own (or others') investigations. Critical ancillary data, such as gene expression data, are usually shared at time of publication, but there is a paucity of data sharing outside the realm of publications and it is usually limited to large consortia (ENCODE, TCGA), or government-mandated data sharing (data.gov). The National Institutes of Health and National Science Foundation both pass data-sharing mandates on to their awardees, but leave the implementation of those mandates to the awardees. An easy solution to data sharing would help federal grantees comply with award requirements and also help create more open, shareable data resources. Additionally, from our own experience there is a wealth of data that is rarely shared, such as ancillary data that does not make it into publications, negative findings,

and findings from investigations that were not fully completed due to resource issues. Many institutions lack the expertise to transform local data into accepted data standards. There are also data that are ready to be shared (such as lists of specimens, and annotations), but few institutions have the technical means to host it for others using a grid-enabled system.

Efforts to facilitate data sharing are common, but few are truly successful. We believe that most data sharing initiatives do not adequately address two key ingredients for a working data sharing environment: few constraints on how to share data, and a recognized avenue for receiving academic recognition (such as recognized citations). Most data sharing initiatives are built on data standards, which promise seamless data exchange at the expense of flexibility. Such initiatives (such as caBIG [1]) can also be overly technical without offering avenues for straightforward data sharing. Finally, few initiatives specify how academic credit is established for shared content. One reason that the scientific community is not sharing data fully is that there are no commonly accepted standards to publish and cite researchers' data-level contributions. We propose a new mode of data-sharing that we believe will be successful for the following two major reasons: First, the use of natural language provides a low barrier to entry for authors to express their research findings; and second, authors value publications as they offer the standard accepted proof of their academic work.

Towards this end, we are building a data sharing infrastructure with the following key features: first, a flexible data sharing setup, which allows for the sharing of plain text, excel, and other similar documents, with the ability to gracefully add metadata when needed; and second, the use of nanopublications, tiny and highly standardized statements that are useful for establishing provenance and academic credit, and for expressing high-level insights into the shared data. Our architecture is built upon Semantic Web technology, and is thus compatible with existing linked data sharing efforts.

Our infrastructure, called Prizms, is built entirely on open source software, leveraging existing data exchange software such as CKAN.<sup>1</sup> We have deployed instances of CKAN and Prizms at [melagrid.org](http://melagrid.org) to serve the SPORE in skin cancer institutes to sharing melanoma related data.<sup>2</sup> The SPOREs have an active data sharing culture, and have recognized the need for exchanging research information. We are using the Prizms infrastructure ([lod.melagrid.org](http://lod.melagrid.org)) to extend the existing MelaGrid data portal ([data.melagrid.org](http://data.melagrid.org)), used for sharing SPORE-related data. To encourage the use of [data.melagrid.org](http://data.melagrid.org) by the melanoma community, we have populated it with melanoma-related datasets from ArrayExpress using a CKAN harvester we developed.<sup>3</sup> We currently have over 331 datasets in our repository.

The Prizms architecture leverages the Linked Data philosophy: use identifiers for things (URLs) that are addresses where consumers can get more information. When a human visits that address, they get a human-readable web page, with useful information, visualizations, and links to other resources. When a machine

---

<sup>1</sup> <http://ckan.org>

<sup>2</sup> <http://trp.cancer.gov/spores/skin.htm>

<sup>3</sup> <https://github.com/jimmccusker/ckanext-arrayexpress>



visits the page, it gets an RDF representation of the thing identified by the URL. The RDF should re-use existing resources that also follow the Linked Data philosophy, thereby providing aggregate benefits to both resources [2]. We will show how we provide a simple means of dataset discovery and citation for scientists and present a framework we use, composed of proven semantic technologies, to provide on-demand enhancement of that data into high-quality Linked Data.

## 2 Requirements: Levels of Data Sharing

Our experience suggests that only a few basic levels of data description are needed to promote successful data sharing. We want to make the value received from data description to be at least linearly related to the effort put into that description, and we want the value to pay off even at very simplistic levels of description. We therefore propose 5 levels of data sharing that will take data providers from very little effort (Level 1) to fully integrated and semantically enriched data that is easy to discover, integrate, and use (Level 5). Each of these levels serves as a broad use case for data sharing based on increasing levels of sophistication.

**Level 1: Basic data sharing** Basic data sharing consists of users 1) posting data somewhere, 2) telling the world about it (such as where it is, when it was modified, who controls it, or a simple description to make it more searchable). This information, often called provenance [3], consists of the basic information about data, such as who controls it, what is it about, when was it created, where can one get it, why was it created, and how was it created and used?

**Level 2: Automated Conversion** Using no domain knowledge, tools can create “naive”, or non-knowledge driven, conversions of tabular data into structured formats such as RDF to provide basic search, browsing, and data integration.

**Level 3: Semantic enhancement** Semantic enhancement is performed using tools that allow users to specify improved data representations beyond what a computer can provide without additional knowledge. This can be by the data originator or other parties.

**Level 4: Semantic eScience** Further annotation and enhancement can be performed by describing the metadata for the dataset using vocabularies with well understood semantics. This provides a foundational component of Semantic E-Science, and corresponds to caBIG-style data sharing.

**Level 5: Community-Based Standards** By providing a framework for communication and discovery of consensus ontology use, a system can assist communities to converge on standard representations of data that result in interoperability across organizations. Further, by giving credit to contributors, the system can make it easier to find a community member that is able to assist in data representation challenges, which enables content-oriented collaborations among geographically or organizationally disparate community members.

### 3 Nanopublications for Datasets: Datapubs

MelaGrid reuses the existing open-source cataloging system CKAN to list and describe publishers' datasets. CKAN accounts for a majority of the basic Level 1 data sharing information that we identify in the previous section. However, it is incomplete, only providing information about dataset publication dates, data locations and hosting, but does not provide a means to describe how the data was produced, nor does it provide a sophisticated mechanism for identification of data owners. We have extended the CKAN RDF publication template to make better use of the available metadata in CKAN using DCAT, DC Terms, and PROV-O. This generates a novel form of nanopublication [4] we call a datapublication, or datapub. We have also included an interface (see Figure 1) that makes it easy to cite published datasets using plain text for non-technical users such as biologists and clinical researchers, BibTeX, PROV, or direct use of a nanopublication [4]. This functionality is available as an Open Source CKAN extension in GitHub called `ckanext-datapub`.<sup>4</sup> We have manually uploaded a dataset from a recent publication [5] and have cited it here using BibTeX. All citation modalities, including plain text, provide a Linked Data URL that provides human and machine-readable representations of the dataset using content negotiation.

Text BibTeX Nanopublication

To get this dataset entry as a nanopublication, use **content negotiation** to request this URL as **TriG** (`application/x-trig`) or use this URL:

`http://data.melagrid.org/dataset/exome-variants-in-melanoma.trig`

This dataset is also available in **Turtle** (`text/turtle`) using content negotiation or using this URL:

`http://data.melagrid.org/dataset/exome-variants-in-melanoma.ttl`

Citing this dataset in **PROV-O** is simple, and is already supported using tools like **Taverna**. To state that a dataset is derived from this one, add this assertion to its description (shown here in Turtle):

```
@prefix prov: <http://www.w3.org/ns/prov#> .

<my-dataset-uri> prov:wasDerivedFrom <http://data.melagrid.org/dataset/exome-variants-in-melanoma> .
```

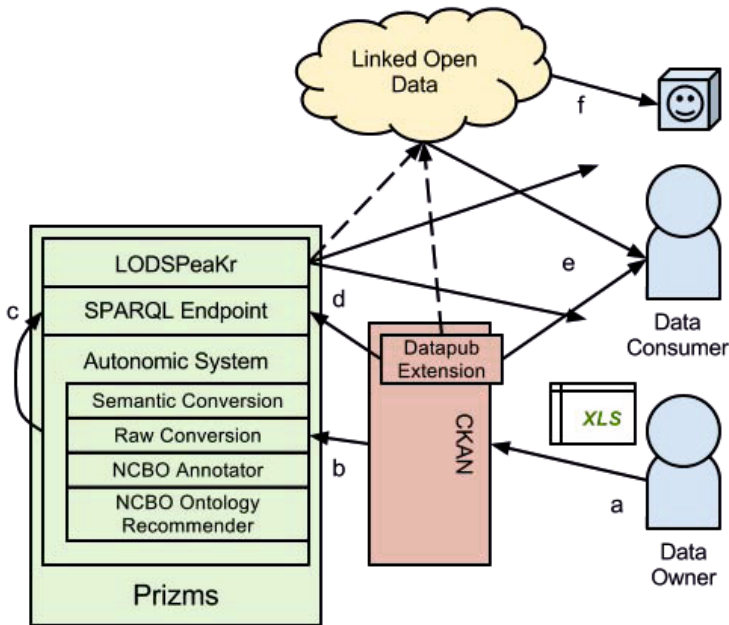
Additional provenance, like attribution, what transformations occurred, etc. can be expressed using additional assertions from PROV-O.

**Fig. 1.** Citing a datapub dataset using plain text, BibTeX, or PROV

<sup>4</sup> <https://github.com/jimmccusker/ckanext-datapub>

## 4 The Prizms Architecture

The Prizms architecture provides the technical foundation to support the remaining four levels of data sharing that we outline above. Prizms combines tools that the Tetherless World Constellation has developed during the past several years for use both internally and externally in many semantic web applications of scientific domains, such as a population science project that integrated health data, tobacco policy, and demographic data [6] and a system for the HHS Developer Challenge developed to integrate a wide variety of health data. The overall workflow of how MelaGrid uses the Prizms architecture and the Datapub extension is shown in Figure 2.



**Fig. 2.** Data flow through Prizms. A data owner (a) submits a dataset to a CKAN instance. This data can be in any format, including Excel (shown), CSV, XML, JSON, or other formats. The Prizms Autonomic System (b) recognizes the addition or change of a dataset and triggers tools that are “interested” in particular new datasets. It is then hosted by a standards-compliant SPARQL endpoint (c). The datapub CKAN extension then (d) generates RDF to describe the dataset as a Datapub. Human data consumers (e) can then browse the full dataset listing and access the data itself using either the traditional CKAN web interface or a Linked Data portal created using LODSPeaKr. Computational data consumers (f) can then access the data in conjunction with the Linked Open Data ecosystem.

While MelaGrid uses CKAN with the Datapub extension to address Level 1 “Basic” data sharing requirements, Prizms exposes the essential data access information as Linked Data using the W3C’s Dataset CATalog vocabulary (DCAT),<sup>5</sup> the Dublin Core Terms (DC Terms) vocabulary,<sup>6</sup> and the W3C’s PROV-O [7] provenance ontology. Prizms addresses Level 2 data-sharing requirements (automated RDF conversion) by using the access metadata to retrieve, organize, and automatically translate data posted to CKAN (such as Excel files) into RDF data files and hosting portions of each in a publicly-accessible SPARQL endpoint. All processing steps record a wealth of provenance described in best practice vocabularies such as Dublin Core, VoID,<sup>7</sup> and PROV-O, which enables transparency of any of Prizms’ data products. For example, any RDF triple or RDF file can be traced back to the original data file(s) and the original publisher(s) [8]. This is important to maintain the reputability of Prizms, which serves as a third party integrator of others’ data.

Prizms addresses Level 3 data-sharing (semantic enhancement) by transforming the original data to user-defined RDF. In the case of tabular data, such as Excel or CSV, transformations are specified using a domain-independent declarative description which itself is encoded in RDF. For example, one can specify that the third column in the data is mapped to a user-specified RDF class for concepts like gender or diagnosis. These concise transformation descriptions can be shared, updated, repurposed, and reapplied to new versions of the same dataset or within other instances of Prizms; they can also be maintained on code hosting sites like GitHub.com or Google Code. The transformation descriptions also serve as additional metadata that can be included as part of queries for the data (e.g., finding all datasets that were enhanced to use the class “specimen”).

Reusing existing entities and vocabularies is the heart of Level 4 data-sharing (Semantic eScience), and using community-agreed ontologies and vocabularies are essential to Level 5 data sharing. We use new parameters of the same semantic conversion tools that are described in Level 2 for this purpose. In addition, datasets can be automatically augmented to produce inferences based on well-structured data that appears in Prizms’ data store. For example, Prizms will augment any address encoded using the vCard RDF vocabulary<sup>8</sup> with the corresponding latitude and longitude (which it computes using the Google Maps API). When clients request Prizms’ data elements, Prizms includes links to other available datasets based on a variety of curated and heuristic connections. These link suggestions can motivate community effort to mature the data towards more mature levels of data sharing.

At all levels of data-sharing, Prizms uses the LODSPeaKr web framework to create Linked Data applications and publish RDF data quickly and with minimal effort. LODSPeaKr provides a set of functionalities that not only improves the accessibility of the data for humans but also for machines by providing content

---

<sup>5</sup> <http://www.w3.org/TR/vocab-dcat/>

<sup>6</sup> <http://purl.org/dc/terms/>

<sup>7</sup> <http://www.w3.org/TR/void/>

<sup>8</sup> <http://www.w3.org/Submission/vcard-rdf/>

negotiation (i.e, the ability to return different formats depending on the client's request for the data element URL). This increases accessibility of the data while minimizing the workload for the development team. Additionally, the system allows the creation of new web pages to display particular subsets of the data that users may considered important. Data consumers can also perform query operations against the backing SPARQL endpoint.

## 5 Discussion

The MelaGrid initiative provides usable, integrated informatics systems that enable collaboration, data sharing, and enhanced analysis to research groups studying skin cancer. Specimen and associated Omics data sharing is a high priority for the MelaGrid initiative. Clinical annotations and phenotyping of specimens, along with Single Nucleotide Polymorphism (SNP), transcription, methylation, and copy number are just a few of the types of data that have become important in cancer research. All of these data have representation in the ArrayExpress subset of data.melagrid.org, and we will be extending its use with additional information from tools like caTissue.

The consortium's first priority is to increase the number of shared data entities, and Prizms's flexible architecture is assisting in this goal. Melagrid has the support of all four national skin SPOREs for use of this infrastructure. Currently, all shared data is at Level 1 (raw data with associated datapubs), and Level 2 (automatic RDF conversion). We will be using the Prizms architecture for converting institution-specific data descriptions into an accepted SPORE OWL/RDF Ontology (currently CDEs, as defined on melagrid.org) as appropriate. This is Level 5 data sharing in Prizms, as it involves a community-agreed standard (Level 3 is using a locally developed ontology, and Level 4 is re-use of ontologies, but not necessarily in a community-agreed manner).

## 6 Future Work

Currently, Prizms can be applied to dataset collections with other content domains, and it offers the same benefits that MelaGrid provides for melanoma data. We look forward to developing Prizms as we apply it to other applications, and we expect that others will find value by doing the same. For example, we are starting a portal for clinical depression treatment based on the Prizms infrastructure. Because using CKAN and the Datapub extension with Prizms has been so useful, we expect to extend Prizms to include both of them in future versions, so that we can facilitate others' adoption of all three components. We also look forward to developing additional out-of-the box capabilities for any datasets that Prizms is used to integrate, such as better connected exploration, better overviews, and better recommendations or guidance on how the data could be better modeled using best practice modeling techniques.

## 7 Conclusion

We have described an infrastructure for creating and using next generation science data portals. We have used the infrastructure to create two data portals - one reported on here in melanoma data and one in response to the human health services data challenge.<sup>9</sup> We have described how our infrastructure supports assimilating, publishing, and enhancing science data into best practices formats. The CKAN infrastructure makes it easy to aggregate data from multiple sources through its harvester framework and we have developed and used a CKAN harvester to obtain and populate data.melagrid.org with 330 melanoma datasets that are now published as linked data. Further, we have provided a citation method for people to cite datasets from within both publications and subsequently-derived datasets using the emerging nanopublication (via our use of datapubs) and World Wide Web Consortium provenance standards.

## References

1. von Eschenbach, A.C., Buetow, K.: Cancer informatics vision: caBIG. *Cancer Informatics* 2, 22–24 (2006)
2. Berners-Lee, T.: *Linked Data - Design Issues* (2006)
3. 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, pp. 316–330. Springer, Heidelberg (2000)
4. Groth, P., Gibson, A., Velterop, J.: The anatomy of a nanopublication. *Information Services and Use* 30(1), 51–56 (2010)
5. Krauthammer, M.: Variant data from "exome sequencing identifies recurrent somatic *rac1* mutations in melanoma", <http://data.melagrid.org/dataset/exome-variants-in-melanoma> (last accessed: February 16, 2013)
6. McCusker, J.P., McGuinness, D.L., Lee, J., Thomas, C., Courtney, P., Tatalovich, Z., Contractor, N., Morgan, G., Shaikh, A.: Towards Next Generation Health Data Exploration: A Data Cube-based Investigation into Population Statistics for Tobacco. In: *Proceedings of the Hawaii International Conference for System Science* (2013)
7. Lebo, T., Sahoo, S., McGuinness, D.: PROV-O: The PROV Ontology, <http://www.w3.org/TR/prov-o/>
8. Lebo, T., Wang, P., Graves, A., McGuinness, D.L.: Towards unified provenance granularities. In: Groth, P., Frew, J. (eds.) *IPAW 2012*. LNCS, vol. 7525, pp. 39–51. Springer, Heidelberg (2012)

---

<sup>9</sup> <http://healthdata.tw.rpi.edu>

# Putting It All Together: The Design of a Pipeline for Genome-Wide Functional Annotation of Fungi in the Modern Era of “-Omics” Data and Systems Biology

Greg Butler

Department of Computer Science and Software Engineering,  
Concordia University, Montreal, Canada  
`gregb@cs.concordia.ca`

**Abstract.** The context for bioinformatics continues to change as new technology brings more varied data in greater volume. We present the preliminary design of a pipeline for functional annotation of fungal genomes. Genome-wide functional annotation benefits from the variety and volume of data available from “-omics” technology, and benefits from the perspective of systems biology.

## 1 Introduction

The outcome of genomics-related research rests on the quality of the genome assembly, the gene predictions, and the functional annotation of the genes. Advances in biotechnology are providing the volume, variety, and quality of data to greatly improve the quality of assemblies, gene models, and functional annotations, and allow improved analysis of expression data and the construction of models for systems biology.

The annotation of prokaryote genomes is mature [1] compared to eukaryote genomes: they are smaller, simpler, and many have been sequenced. Genome assembly, structural annotation, and functional annotation for eukaryotes is relatively less mature. Here we focus on fungal genomes, as they are our area of expertise, and are the simplest starting point amongst the eukaryotes. Fungi are important but often overlooked organisms that affect our daily lives as causative agents of disease, as sources of food, as agents for recycling of biomass, and as key ingredients in industrial processes. The information contained in their genomes can enhance our understanding of how they function, and on their uses and impacts. There is no open-source software that embodies a comprehensive pipeline for structural and functional annotation of a fungal genome, thereby limiting the access of genomics by the fungal research community.

In this paper, we look at the problem of functional annotation and how to utilise the available data for a species about the genome, transcriptome, and proteome for that purpose. We consider the problem of functional annotation in the context of the existing databases of gene annotations, and other resources, as

well as in the context of modern techniques such as network analysis, metabolic pathway reconstruction, and systems biology.

Functional annotation generally focuses on protein functional prediction from the protein sequence, though predictions may also use protein structural information when available. The reviews [2, 3] clearly explain the difficulty of protein function prediction, the types of errors that are common, and the concerns about the quality of existing annotations in databases. The difficulty is inherent due to the multifunctional roles of proteins, the potential binding with many substrates, and the significant impact that a change of a single amino acid can have on structure and function. There are many roles for functional annotation: (1) Collecting evidence for manual curation; (2) Understanding the role(s) of a gene; (3) Understanding the roles and relationships of a set of genes; (4) Understanding the gene complement of the organism; (5) Forming the basis for the analysis of expression data, metabolic reconstruction, and system models; and (6) Forming the basis for publications about the genome through stories about specific genes, or interesting comparisons across several organisms. So a functional annotation pipeline must do more than associate a list of GO terms with each gene [4]. We augment GO term annotations with phylogenomic classification into protein families and subfamilies as in PANTHER [5], and include predicted protein domain architecture, secondary structure, and post-translational modifications sites as in ANNIE [6], and construct a feature space and a set of gene networks for data mining.

## 2 Background

Major centres have published brief descriptions of their procedures for functional annotation of fungal genomes: JGI [7], Broad [8], MIPS [9], and Génolevures [10]. None has made their pipelines available. IGS has published their standard operating procedure (SOP) [4] for functional annotation of prokaryote genomes and released the pipeline as part of the CloVR system [11]. Functional annotation [3, 8, 12–16] is performed by gathering evidence of function from various sources, integrating the evidence, and presenting the inferred function. Evidence is inferred from properties of related sequences using annotation transfer by homology (ATH), guilt-by-association, or other prediction methods. Integration occurs through statistical summary of GO terms, classical machine learning approaches, or network analysis [16]. The major steps are: (1) Annotation transfer by homology (ATH) using sequence similarity; (2) Protein family recognition for ATH using Hidden Markov Model (HMM) classifiers, sometimes including resolution of orthologs, paralogs, and protein subfamily membership; (3) Case-based reasoning for specific types of genes and proteins [8, Section 7]: carbohydrate-active enzymes, transporters, transcription factors, effectors, proteases, protein kinases, histidine kinases, G protein-coupled receptors, secondary metabolite gene clusters; (4) Feature recognition for signals, motifs, and domains, and mapping features to annotation terms; in particular, the prediction of localization relies heavily on this approach, as does PRIAM [17] for predicting EC numbers, and InterPro2GO for GO terms; (5) Integration and summary of GO terms



of related sequences; (6) Guilt-by-association using genome context [18] (such as gene clusters, gene fusion, phylogenetic profiles), pathway reconstruction, and network analysis; (7) Post-processing by rules to detect common annotation errors, e.g. the Swiss-Prot HAMAP system [19] for prokaryote annotation, and SAAS [20] and UniRule ([www.uniprot.org/program/automatic\\_annotation](http://www.uniprot.org/program/automatic_annotation)) rules at EBI when annotating TrEMBL from Swiss-Prot; and (8) Post-processing data mining [21–23] to detect anomalies for investigation manually by curators, and to automatically infer rules [20]. This is dependent upon existing sequence databases with annotations, existing databases of HMMs with annotations, and existing predictors for signals, motifs, and domains; ie, a trusted set of data as the foundation for the annotation such as CharProtDB [24] for prokaryotes at JCVI. While we concur with Ross Overbeek that this dependence on a foundation of trusted data links functional annotation with the continued maintenance of the foundation, i.e. that “*curation is forever*” [25], it is not clear that the subsystem approach [26] that is taken for prokaryotes, and is the basis of RAST [1], is appropriate for eukaryotes.

## 2.1 Types of Functional Annotation

The primary annotation is the role of a gene product. A *GO triple* is a triple  $(p, m, c)$  of Gene Ontology (GO) terms indicating the biological process  $p$ , the molecular function  $m$ , and the cellular component  $c$  for the role. A *family pair* is a pair  $(F, f)$  of family  $F$  and subfamily  $f$  for the gene product. A *functional annotation* of a gene consists of a GO triple together with a family pair. The functional annotation described as a GO triple and a family pair allows us to make distinctions in the role of genes that is supported by existing sequence data in the foundation but not yet formalized in the Gene Ontology. Furthermore, it brings in distinctions and terminology, such as glycoside hydrolase (GH) family, which are in common use by genomics researchers but not in the Gene Ontology.

Since many genes cannot be assigned a primary annotation as complete as this, a collection of *features* will be included in the annotation. Features include individual GO terms (that are not part of a GO triple) inferred from various searchers and predictors; EC (Enzyme Commission) numbers; information about gene or protein regions or sites such as signals, binding modules, or catalytic sites, secondary structure, post-translational modifications, and transcription factor binding sites.

A *metabolic pathway reconstruction* is included in the annotation as a Pathway-Genome database (PGDB) as produced by Pathway Tools.

*Gene networks* are constructed as part of the annotation. Each gene network consists of nodes representing genes or proteins, and edges that represent the existence of a relationship or shared property of the pair of genes and/or proteins. Types of gene networks include interaction, shared protein domains, and co-localization networks. Important to this work are co-expression networks that encode the information in the expression data; co-pathway networks derived from the metabolic pathway reconstruction; and networks for comparative genomics relationships such as orthology and synteny.

## 2.2 Resolving Orthology

The application of annotation transfer via homology is prone to errors. Homology is about the evolutionary relationship of genes through the processes of speciation, gene duplication, and horizontal gene transfer. Orthologs are homologous sequences derived from a speciation event, while paralogs are derived from duplication. In general, orthologs are more likely to retain their function, while paralogs more often do not [27]. Therefore, it is important to resolve orthologs and paralogs during comparative genomics using phylogenetic and phylogenomic techniques [27, 28]. Many functional annotation pipelines use databases of orthologous groups of genes. Other pipelines like PANTHER [5] and Sifter 2.0 [29] apply phylogenomics across the board during annotation to help resolve orthologs and paralogs.

## 2.3 Integration of Evidence

Most functional annotation pipelines have a stage to reconcile descriptions and GO terms for a protein that are obtained from the diverse search hits, such as Blast Extend Repraze (BER) [4], Gotcha, Blast2GO, or Ontologizer. These give more weight to strong specific hits but can also treat weaker or more remote hits when that is the only evidence available for a protein. Daisuke Kihara's PFP [30] and ESG [31] showed that, for the analysis of expression data, it is worthwhile to annotate poorly characterized proteins with GO terms which may be high in the GO hierarchy even when there is low confidence in those terms [32]. ESG performs better than PFP, PSI-Blast, Gotcha, and iprscan with InterPro2GO.

## 2.4 Pathway Reconstruction

An important task in systems biology is the modeling of metabolism [33] which uses “-omics” data to fill in gaps in the models [34]. Pathway reconstruction is typified by Pathway Tools [35] which uses the MetaCyc knowledgebase of pathways curated from the literature to provide a template of metabolic, transport, and regulatory pathways against which to match the roles of proteins in a genome. The tools construct a Pathway Genome Database (PGDB). Using an existing functional annotation, the tools first match genes to reactions, then determine whether each pathway is present or not in the organism [36, 37]. The pathways present may have holes, which means that certain reactions in the pathway are not matched with a gene. The hole-filling algorithm [38] uses a Bayesian approach to rank the unmatched genes in the genome with each hole as a potential match, and the software allows curators to accept or reject a match. An accepted match assigns a molecular function (the reaction) and a biological process (the pathway) to the gene. For pathways that are not in MetaCyc, there are approaches to analyse expression data [16] or comparative genomes [39] to discover novel subsystems such as protein complexes and pathways.

## 2.5 Network-Based Annotation

Network analysis [16] is typically applied to co-expression networks as part of advanced analysis of expression data. Network analysis is one approach to integrate diverse evidence. Its usefulness in functional annotation is established [13–16]. GeneMANIA [40] is a system for network analysis that had success in the MousFunc annotation competition.

Graph-based semi-supervised learning represents evidence as (weighted) edges between vertices (i.e. genes or proteins) and as labels on vertices representing annotations. Unlabelled vertices denote a lack of annotation and the goal of machine learning is to propagate annotations to these unlabelled vertices from the set of nearest relevant neighbours. The field began in 2004 at Tübingen [41] solving the problem using Support Vector Machines with graph-specific kernels, but also developing rank propagation algorithms (as in Google’s PageRank), and linear algebra based approaches for a matrix representation of the graph. The linear algebra approach led to fast graph integration [42], and to advances in efficiency in GeneMANIA [43] offering real-time response as a web service. There are two subproblems in graph integration: learning the weights to apply to each graph when combining them into a single graph; and learning the solution for the integrated graph. The GeneMANIA algorithm solves both subproblems at the same time. GeneMANIA also provides tools to gauge the impact of each graph on the analysis. GeneMANIA currently has graphs representing protein and genetic interactions, pathways, co-expression, co-localization, and protein domain sharing. Its format for graph data is generic, so one can encode other data as graphs for use by GeneMANIA. A strength of GeneMANIA is its collection of datasets for model organisms. A weakness is that it only addresses model organisms. To utilise the datasets from *Saccharomyces cerevisiae*, our functional annotation pipeline needs to relate other fungal genomes to that of yeast which we can do in various ways with graphs representing sequence similarity (see SIMAP [44] and STRINGv7 [45]), ortholog and paralog relationships (see eggNOG [46]), or synteny (see STRINGv8 [47]).

## 2.6 Structural Annotation

The impact of good quality gene models percolates throughout the downstream processing since a correct gene model, and therefore CDS, allows further analysis to detect features such as signal peptides, binding sites, protein domains, sites of post-translational modification, and anchors, that might be missed due to truncated C- or N-terminals, missing exons, or missing parts of exons. In proteogenomics [48–50] proteins undergo *de novo* sequencing, or terminal sequencing, and the peptides are mapped to the six-frame translation of the genome.

Gene prediction programs can be classified as *ab initio* or homology-based. *Ab initio* predictors search the genome sequence with Hidden Markov Models to detect signals that indicate coding potential, such as open reading frames, intron donor and acceptor sites, and transcription start and stop sites [51]. Homology-based predictors search for sequences similar to known genes in related

organisms, or sequences that translate to the sequences of known proteins. Practical gene prediction pipelines combine the outputs of multiple predictors with additional evidence [52], including transcriptomics and proteogenomics, which requires that data be mapped to the genome.

### 3 Specification of the Pipeline

For the functional annotation pipeline, the **inputs** are

- the genome assembly;
- the genome defined as a set of genes, as a **gff** file for the gene models, and the corresponding fasta sequence files for genes, CDS, transcripts, proteins, upstream and downstream regions;
- the expression data from transcriptomics and proteomics; and
- reference datasets, classifiers, and predictors, etc.

The **output** produced by functional annotation consists of a set of gene pages (one for each gene model), a summary table of the genes and their annotation, the expression data from transcriptomics and proteomics, a Pathway-Genome Database for the reconstructed metabolic pathways, a set of networks, and a gene feature space for data mining. These can be accessed and viewed by web browsers, Pathway Tools, and Cytoscape. They can be exported to other systems for further analysis, such as Kbase, Galaxy, R/Bioconductor, and Weka.

The gene page presents the primary functional annotation. Each functional annotation identifies the GO triple and the family pair. The gene page includes several organized views of features: the genome context of the gene; the structure of the gene model in terms of introns and exons; the site information for upstream and downstream regions; the protein domain architecture including signals, cleavage sites, binding modules, catalytic domains, and low complexity regions; protein secondary structure; and the sites of post-translational modifications. Features that are not accommodated in the organized views are listed individually. Links are provided to supporting evidence such as alignments and tool output, as well as to the syntenic blocks and orthologous groups to which the gene belongs. The gene pages are in **html**.

A gene table provides a summary annotation of each gene in the genome. This includes an identifier; a descriptive name; the GO triples and family pairs; individual GO terms by category of biological process, molecular function, and cellular component; EC number; and InterPro domains. The gene table in **html** acts as an index into the gene pages. For convenience, the gene table is also provided as a tab-separated value (**tsv**) file. Furthermore, the common information on GO annotation terms, EC numbers, and InterPro results are provided as separate files in the standard formats. An extended gene table is provided as a **tsv** file and contains the organised views (encoded as readable text), and information on orthologous clusters, and on the phylogenetic distribution of homologs across the foundation fungal genomes, the model organisms, the fungal phyla, and the kingdoms of life.

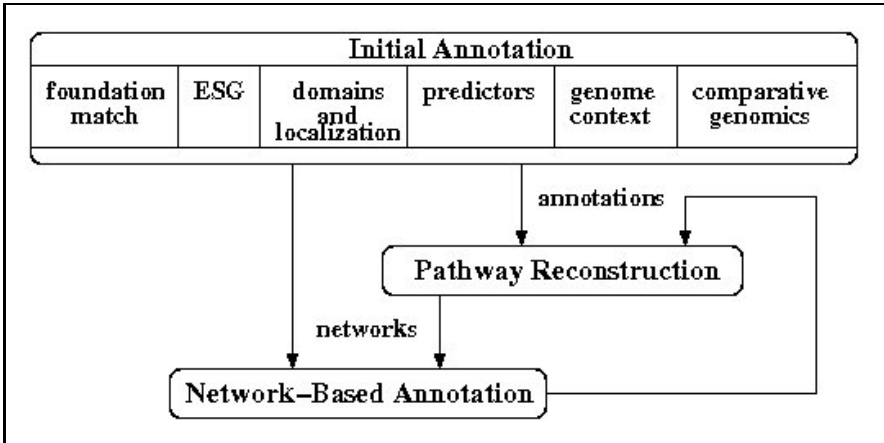


Fig. 1. Functional Annotation Stages

The gene feature space consists of an extensive set of features and classifications for the purpose of machine learning and data mining. Each gene has a vector of atomic features consisting of numbers or identifiers. There are standard approaches to encode information such as sequence composition, and annotation terms in classification schemes like EC, GO, and InterPro whether they be hierarchical or multi-valued.

## 4 Design of the Pipeline

The functional annotation process consists of three main stages. The first stage is the initial annotation; the second stage is metabolic pathway reconstruction; and the third stage is network-based annotation. The second and third stages can be repeated as further new annotations may be produced. The first stage consists of a number of steps, each of which is independent of the others. A wide variety of tools are run in order to produce features (including individual GO terms and EC numbers), and networks. Functional annotation in the sense of a GO triple and a family pair are only produced from strong matches to the foundation data by applying rules for annotation transfer by homology from the foundation data annotations.

### 4.1 Stage One: Initial Annotation

The first stage can be viewed as six parts: part one matches the proteins to the foundation data; part two applies the extended similarity group (ESG) method to the proteins; part three performs major tasks of applying iprscan against InterPro, and predicting localization; part four applies numerous useful predictors, including PRIAM, to detect features; part five considers the genome context information; and part six considers comparative genomics information including orthology and synteny.

**Table 1.** Outputs of the Stages of Functional Annotation

Stage	Part	Tool	Outputs					
			GO triple	Family pair	GO term	Features other	Network type	Pathway PGDB
1	1	Foundation Match	Y	Y				
1	2	ESG			Y			
1	3	InterPro localization			MF CC	Y Y	shared domain co-localization	
1	4	predictors PRIAM			MF	Y EC		
1	5	Genome context				Y	co-chromosomal	
1	6	Comparative genomics					similarity orthology synteny	
2		Pathway Tools	Y				co-pathway	Y
3		GeneMANIA			Y			

Part one matches proteins in the target genome against the foundation using similarity searches, hmmer3 searches against Hidden Markov Models (HMM) of families and subfamilies, and specialized predictors for specific families. Each sequence in the foundation is associated with a trusted functional annotation as a GO triple and family pair. Each family and subfamily is associated with an annotation that is as specific as possible while still being conserved across all members of the family or subfamily. The general rule for annotation transfer by homology is that when there is a strong match between a protein in the target genome and a sequence, family or subfamily, then transfer the associated annotation. Rules to handle exceptions to the general rule are applied.

Part two matches each protein in the target genome against sequence databases in the foundation and against external datasets (supplied as a parameter) using the extended similarity group method (ESG) [31] to summarize GO terms from matches to poorly annotated proteins. It is more sensitive than PSI-Blast. For the analysis of expression data, it is worthwhile to annotate poorly characterized proteins with GO terms which may be high in the GO hierarchy even when there is low confidence in those terms [32].

Part three matches each protein in the target genome against the InterPro database using iprscan to produce a set of protein domain features for the protein. Corresponding individual GO terms are produced using the InterPro2GO mapping. A network based on shared domains is also produced. Part three also runs a series of tools related to the prediction of localization on each protein and produces a set of features, an individual GO term for its cellular component, and a co-localization network. The tools used include SignalP 4.0 [53], Phobius [54], TargetP [55], TMHMM [56], WolfPSORT [57], and MultiLOC2 [58].

Part four runs PRIAM [17] on each protein to produce an EC number as a feature. Part four also runs a series of predictors for secondary structure; low complexity regions; post-translational modifications; GPI anchors; promoter binding sites, etc; to produce features of regions and sites.

Part five considers genome context for the target genome. For each gene it computes a gene copy number counting the number of paralogs in the genome. Part five also produces a genome context network where there is an edge between two genes if and only if the genes lie on the same chromosome (or scaffold).

Part six considers comparative genomics. Part six performs similarity searches against genomes in the foundation and for model organisms to produce a similarity network. Part six also runs a series of predictors for orthologous groups such as eggNOG [46], orthoMCL [59], InParanoid [60], and OMA [61] to produce various orthology networks, including one based on reciprocal best hits (RBH). Part six also runs tools for determining synteny [62, 63] between the target genome and those genomes in the foundation and for model organisms to produce several synteny networks where there is an edge between two genes in the target genome if and only if they lie in same syntenic block.

## 4.2 Stage Two: Metabolic Pathway Reconstruction

The second stage is metabolic pathway reconstruction done by Pathway Tools. The input consists of the annotation of each gene in terms of a text description, GO terms, and EC numbers. The output consists of the assignment of genes to reactions in pathways for both metabolic reactions and transport reactions. This assignment provides a GO triple as an annotation for the gene. This stage also produces a co-pathway network; and a Pathway-Genome Database. When the second stage is repeated following the third stage of network-based annotation, then the metabolic pathway reconstruction only requires updating rather than re-computation. Following the updates to the annotation, the hole-filling methods in Pathway Tools may assign further genes to reactions.

Pathway Tools [35] uses the MetaCyc knowledgebase of pathways that have been curated from the literature to provide a template of metabolic, transport, and regulatory reactions and pathways. Using an existing functional annotation, the tools first match genes to reactions, then determine whether each pathway is present or not in the organism [36, 37]. The pathways present may have holes; that is, there are orphan reactions in the pathway that are not assigned to a gene. The hole-filling algorithm [38, 64] uses a Bayesian approach to rank the unassigned genes in the genome with each hole, and the software allows curators to accept or reject a match. There are alternative hole-filling approaches that use orthology (AutoGraph [65]) and expression data (GLOBUS [66]). The use of orthology should be redundant here in our pipeline because of our inclusion of metabolic pathway reconstructions in the foundation. Similarly, the use of expression data should be redundant because network-based annotation will use expression data to assign genes to molecular function GO terms. We need to determine appropriate thresholds for when to transfer an annotation for ranked candidate genes to fill a hole automatically.

### 4.3 Stage Three: Network-Based Annotation

The third stage is network-based annotation done by GeneMANIA [40]. The input is the set of networks produced by the first and second stages. The output is new gene annotations as GO terms, or an increase in confidence for an existing GO term annotation of a gene. The third stage also provides integration of the evidence for functional annotation of each gene.

GeneMANIA currently has graphs representing protein and genetic interactions, pathways, co-expression, co-localization, and protein domain sharing. Its format for graph data is generic, so one can encode other data as graphs for use by GeneMANIA. The datasets in GeneMANIA includes the yeast *S. cerevisiae*. Information from prior steps will be encoded as graphs representing pathways, co-expression, co-localization, and protein domain sharing in the target genome.

GeneMANIA is used to study a single model organism. We need to study non-model organisms, yet benefit from the available datasets for model organisms. Hence we propose to develop new types of gene networks representing sequence similarity [44, 45], ortholog and paralog relationships [46], or synteny [47]. These networks connect the genes of target genomes to genes in existing datasets.

### 4.4 Foundation

The foundation data for fungi will include experimentally characterized and trusted data. It will be built out in stages: (a) lignocellulose-active proteins [67]; (b) the manually curated genes and annotations of *S. cerevisiae*, *S. pombe*, *C. albicans*, *N. crassa*, *A. nidulans*, and *A. fumigatus*; (c) Swiss-Prot; (d) tRNA genes and tRNA synthetase genes; (e) the metabolism reconstructions of fungi, including transporters, from the BioMet collection [68]; (f) transcription factors in fungi; (g) genes related to the secretory pathway; (h) genes related to cell wall biosynthesis; and (i) fungal core genes.

In the construction of the foundation, we perform clustering, multiple sequence alignment, and build HMMs for well-understood families so as to resolve orthologs and paralogs, and thereby sidestep the general use of phylogenomics. Subfamilies are determined using SciPhy [69], and Secator and DPC [70]. It is for such well-understood families that rules governing transfer of annotations were developed for prokaryotes in HAMAP.

### 4.5 Rules, Anomalies, and Data Mining

Rules are implemented for annotation transfer by homology (ATH) first by identifying the functional annotation (as a GO triple and family pair) that is conserved across a family or subfamily amongst the well-understood protein families in the foundation. Exceptions to ATH in this context may arise based on phylogenetic variation between the foundation proteins and the target genome, in which case they will be captured by additional rules (for the exception) as in HAMAP (for prokaryotes). Rules will also be developed when curators notice common errors in annotation made by the tools or pipelines. The conditions



under which those errors occur may be identified by mining the gene feature space; if so, a rule can be developed to catch such errors. The rules can be applied as post-processing steps. The rules need to relate available annotated features of proteins, together with phylogenetic location of the fungi, to the functional annotation (primarily GO terms) and errors in functional annotation. Furthermore, there are machine learning techniques for anomaly detection that can be applied to the gene feature space. Anomalies raise concerns about the accuracy of annotations that can be investigated by the curators.

Features include sequence composition (EMBOSS pepstats generates 61 features, ExPasy ProtParam generates 36 features); 37955 GO terms (v1.3418); 23232 Interpro entries (v38.0); membership in families or clusters; and results from the various predictors.

## 5 Benchmarking

Datasets and benchmarking are critical for the development of pipelines. Progress is made during development by examining false positive and false negative results when benchmarking, trying to identify the cause for the errors, and making improvements to the pipeline. When testing on novel genomes, a curator samples a small number (about 100) of results manually to determine which can be classified as false positive or false negative.

The primary datasets for the benchmarking of the functional annotation pipeline are those for *Aspergillus niger*, *Thermomyces lanuginosus*, and *Phanerochaete chrysosporium*, that were used for our structural annotation pipeline. We have curated functional annotations for some genes but not for all genes.

We use *Candida albicans* and *S. cerevisiae*, as “gold standard” datasets, taking care to remove related entries from the foundation data that is the basis for functional annotation. In cases where the benchmarking shows differences between the result of the pipeline and the gold standard, we can refer to the yeast biologists at the Centre for Structural and Functional Genomics for adjudication.

## 6 Conclusions

Structural and functional annotation of a genome are basic steps in genomics-related research. It is important that these tasks produce quality results, in terms of coverage of the genome and accuracy of the annotations. The incompleteness of functional annotation of genomes, even for model organisms, is a significant gap in our knowledge. The inaccuracies in annotation databases lead to propagation of erroneous annotations to new genomes which poses a major threat to the validity of any ensuing analysis of experimental data. It is essential that we advance the coverage and accuracy of structural and functional annotation.

This paper presents the design of a pipeline for functional annotation tailored for fungal genomes. The pipeline uses input data from genomics (DNA reads), transcriptomics (RNA reads), and proteomics (mass spectrometric spectra). The pipeline integrates pathway reconstruction, network analysis, and rule-based post-processing. It builds on a foundation of trusted annotations to derive

detailed functional annotations of the role of a gene product where possible, and to provide a broad set of evidence as features, expression data, and networks for further analysis by other tools for system biology and “-omics” data analysis.

**Acknowledgements.** Funding in part provided by Genome Canada, Genome Quebec, and NSERC.

## References

1. Aziz, R.K., Bartels, D., Best, A.A., DeJongh, M., Disz, T., Edwards, R.A., Formsma, K., Gerdes, S., Glass, E.M., Kubal, M., Meyer, F., Olsen, G.J., Olson, R., Osterman, A.L., Overbeek, R.A., McNeil, L.K., Paarmann, D., Paczian, T., Parrello, B., Pusch, G.D., Reich, C., Stevens, R., Vassieva, O., Vonstein, V., Wilke, A., Zagnitkos, O.: The RAST server: rapid annotations using subsystems technology. *BMC Genomics* 9, 75 (2008)
2. Friedberg, I.: Automated protein function prediction—the genomic challenge. *Brief. Bioinform.* 7(3), 225–242 (2006)
3. Erdin, S., Lisewski, A.M., Lichtarge, O.: Protein function prediction: towards integration of similarity metrics. *Curr. Opin. Struct. Biol.* 21(2), 180–188 (2011)
4. Galens, K., Daugherty, S., Creasy, H.H., Angiuoli, S., White, O., Wortman, J., Mahurkar, A., Giglio, M.G.: The IGS standard operating procedure for automated prokaryotic annotation. *Stand. Genomic Sci.* 4(2), 244–251 (2011)
5. Mi, H., Muruganujan, A., Gaudet, P., Lewis, S., Thomas, P.D.: PANTHER version 7: improved phylogenetic trees, orthologs and collaboration with the Gene Ontology Consortium. *Nucleic Acids Res.* 38, D204–D210 (2010)
6. Ooi, H.S., Kwo, C.Y., Wildpaner, M., Sirota, F.L., Eisenhaber, B., Maurer-Stroh, S., Wong, W.C., Schleiffer, A., Schneider, G.: ANNIE: integrated de novo protein sequence annotation. *Nucleic Acids Res.* 37, W435–W440 (2009)
7. Martinez, D., Grigoriev, I.V., Salamov, A.A.: Annotation of fungal genomes. *Proc. ANAS (Biol.)* 65(5-6), 177–183 (2010)
8. Haas, B.J., Pearson, M.D., Cuomo, C.A., Wortman, J.R.: Approaches to fungal genome annotation. *Mycology* 2(3), 118–141 (2011)
9. Mewes, H.W., Frishman, D., Gregory, R., Mannhaupt, G., Mayer, K.F., Münsterkötter, M., Ruepp, A., Spannagl, M., Stümpflen, V., Rattei, T.: MIPS: analysis and annotation of genome information in 2007. *Nucleic Acids Res.* 36, D196–D201 (2008)
10. Martin, T., Durrens, P.: Génolevures: Policy for automated annotation of genome sequences, <http://www.pasteur.fr/ip/resource/filecenter/document/01s-00004f-0e5/abstract-156.pdf>
11. Angiuoli, S.V., Matalka, M., Gussman, G., Galens, K., Vangala, M., Riley, D.R., Arze, C., White, J.R., White, O., Fricke, W.F.: CloVR: A virtual machine for automated and portable sequence analysis from the desktop using cloud computing. *BMC Bioinformatics* 12, 356 (2011)
12. Frishman, D.: Protein annotation at genomic scale: the current status. *Chem. Rev.* 107(8), 3448–3466 (2007)
13. Hawkins, T., Kihara, D.: Function prediction of uncharacterized proteins. *J. Bioinform. Comput. Biol.* 5(1), 1–30 (2007)
14. Janga, S.C., Moreno-Hagelsieb, G.: Network-based function prediction and interactomics: the case for metabolic enzymes. *Metab. Eng.* 13(1), 1–10 (2011)
15. Watson, J.D., Laskowski, R.A., Thornton, J.M.: Predicting protein function from sequence and structural data. *Curr. Opin. Struct. Biol.* 15(3), 275–284 (2005)

16. Sharan, R., Ulitsky, I., Shamir, R.: Network-based prediction of protein function. *Mol. Systems Biol.* 3, 88 (2007)
17. Claudel-Renard, C., Faraut, T., Kahn, D.: Enzyme-specific profiles for genome annotation: PRIAM. *Nucleic Acids Res.* 31(22), 6633–6639 (2003)
18. Ferrer, L., Dale, J.M., Karp, P.D.: A systematic study of genome context methods: calibration, normalization and combination. *BMC Bioinformatics* 11, 493 (2010)
19. Lima, T., Coudert, E., Keller, G., Michoud, K., Rivoire, C., Bulliard, V., de Castro, E., Lachaize, C., Baratin, D., Phan, I., Bougueleret, L., Bairoch, A.: HAMAP: a database of completely sequenced microbial proteome sets and manually curated microbial protein families in UniProtKB/Swiss-Prot. *Nucleic Acids Res.* 37, D471–D478 (2009)
20. Kretschmann, E., Apweiler, R.: Automatic rule generation for protein annotation with the C4. data mining algorithm applied on SWISS-PROT. *Bioinformatics* 17(10), 920–926 (2001)
21. Yu, G.X.: Ruleminer: a knowledge system for supporting high-throughput protein function annotations. *J. Bioinform. Comput. Biol.* 2(4), 615–637 (2004)
22. Artamonova, I.I., Gelfand, M.S., Frishman, D.: Mining sequence annotation data-banks for association patterns. *Bioinformatics* 21, iii49–iii57 (2005)
23. Poptsova, M.S., Gogarten, J.P.: Using comparative genome analysis to identify problems in annotated microbial genomes. *Microbiology* 156(7), 1909–1917 (2010)
24. Madupu, R., Dodson, R.J., Brinkac, L., Harkins, D., Durkin, S., Shrivastava, S., Sutton, G., Haft, D.: CharProtDB: a database of experimentally characterized protein annotations. *Nucleic Acids Res.* 40, D237–D241 (2012)
25. Overbeek, R., Devine, D., Vonstein, V.: Curation is forever: comparative genomics approaches to functional annotation. *Targets* 2(4), 138–146 (2003)
26. Overbeek, R., Begley, T., Butler, R.M., Choudhuri, J.V., Chuang, H.Y., Cohoon, M., de Crécy-Lagard, V., Diaz, N., Disz, T., Edwards, R., Fonstein, M., Frank, E.D., Gerdes, S., Glass, E.M., Goesmann, A., Hanson, A., Iwata-Reuyl, D., Jensen, R., Jamshidi, N., Krause, L., Kubal, M., Larsen, N., Linke, B., McHardy, A.C., Meyer, F., Neuweger, H., Olsen, G., Olson, R., Osterman, A., Portnoy, V., Pusch, G.D., Rodionov, D.A., Rückert, C., Steiner, J., Stevens, R., Thiele, I., Vassieva, O., Ye, Y., Zagnitko, O., Vonstein, V.: The subsystems approach to genome annotation and its use in the project to annotate 1000 genomes. *Nucleic Acids Res.* 33(17), 5691–5702 (2005)
27. Kuzniar, A., van Ham, R.C., Pongor, S., Leunissen, J.A.: The quest for orthologs: finding the corresponding gene across genomes. *Trends Genet.* 24(11), 539–551 (2008)
28. Kristensen, D.M., Wolf, Y.I., Mushegian, A.R., Koonin, E.V.: Computational methods for Gene Orthology inference. *Brief. Bioinform.* 12(5), 379–391 (2011)
29. Engelhardt, B.E., Srouji, J.R., Brenner, S.E.: Genome-scale phylogenetic function annotation of large and diverse protein families. *Genome Res.* 21(11), 1969–1980 (2011)
30. Hawkins, T., Luban, S., Kihara, D.: PFP: Automated prediction of gene ontology functional annotations with confidence scores using protein sequence data. *Proteins* 74(3), 566–582 (2009)
31. Chitale, M., Hawkins, T., Park, C., Kihara, D.: ESG: extended similarity group method for automated protein function prediction. *Bioinformatics* 25(14), 1739–1745 (2009)
32. Hawkins, T., Kihara, D.: Functional enrichment analyses and construction of functional similarity networks with high confidence function prediction by PFP. *BMC Bioinformatics* 11, 265 (2010)

33. Santos, F., Boele, J., Teusink, B.: A practical guide to genome-scale metabolic models and their analysis. *Methods Enzymol.* 500, 509–532 (2011)
34. Orth, J.D., Palsson, B.Ø.: Systematizing the generation of missing metabolic knowledge. *Biotechnol. Bioeng.* 107(3), 403–412 (2010)
35. Karp, P.D., Krummenacker, M., Latendresse, M., Dale, J.M., Lee, T.J., Kaipa, P., Gilham, F., Spaulding, A., Popescu, L., Altman, T., Paulsen, I., Keseler, I.M., Caspi, R.: Pathway Tools version 13.0: integrated software for pathway/genome informatics and systems biology. *Brief. Bioinform.* 11(1), 40–79 (2010)
36. Karp, P.D., Latendresse, M., Caspi, R.: The pathway tools pathway prediction algorithm. *Stand. Genomic Sci.* 5(3), 424–429 (2011)
37. Dale, J.M., Popescu, L., Karp, P.D.: Machine learning methods for metabolic pathway prediction. *BMC Bioinformatics* 11, 15 (2010)
38. Green, M.L., Karp, P.D.: A bayesian method for identifying missing enzymes in predicted metabolic pathway databases. *BMC Bioinformatics* 5, 76 (2004)
39. Ferrer, L., Karp, P.D.: Discovering novel subsystems using comparative genomics. *Bioinformatics* 27(18), 2478–2485 (2011)
40. Warde-Farley, D., Comes, O., Zuberi, K., Badrawi, R., Chao, P., Franz, M., Grouios, C., Kazi, F., Lopes, C.T., Maitland, A., Mostafavi, S., Montojo, J., Shao, O., Wright, G., Bader, G.D., Morris, Q.: The GeneMANIA prediction server: biological network integration for gene prioritization and predicting gene function. *Nucleic Acids Res.* 38, W214–W220 (2010)
41. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: Thrun, S., Saul, L.K., Schölkopf, B. (eds.) *Neural Information Processing Systems 16*. MIT Press (2004)
42. Tsuda, K., Shin, H.J., Schölkopf, B.: Fast protein classification with multiple networks. *Bioinformatics* 21(suppl. 2), ii59–ii65 (2005)
43. Mostafavi, S., Warde-Farley, D., Grouios, C., Morris, Q.: GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function. *Genome Biology* 9(suppl. 1), S4 (2008)
44. Rattei, T., Arnold, R., Tischler, P., Lindner, D., Stümpflen, V., Mewes, H.W.: SIMAP: the similarity matrix of proteins. *Nucleic Acids Res.* 34, D252–D256 (2006)
45. von Mering, C., 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, D358–D362 (2007)
46. Powell, S., Trachana, K., Roth, A., Kuhn, M., Muller, J., Arnold, R., Rattei, T., Letunic, I., Doerks, T., Jensen, L.J., von Mering, C., Bork, P.: eggNOG v3.0: orthologous groups covering 1133 organisms at 41 different taxonomic ranges. *Nucleic Acids Res.* 40, D284–D289 (2012)
47. Jensen, L.J., Stark, M., Chaffron, S., Creevey, C., Muller, J., Doerks, T., Julien, P., Roth, A., Simonovic, M., Bork, P., von Mering, C.: STRING 8—a global view on proteins and their functional interactions in 630 organisms. *Nucleic Acids Res.* 37, D412–D416 (2009)
48. Armengaud, J.: A perfect genome annotation is within reach with the proteomics and genomics alliance. *Curr. Opin. Microbiol.* 12(3), 292–300 (2009)
49. Renuse, S., Chaerkady, R., Pandey, A.: Proteogenomics. *Proteomics* 11(4), 620–630 (2011)
50. Castellana, N., Bafna, V.: Proteogenomics to discover the full coding content of genomes: a computational perspective. *J. Proteomics* 73(11), 2124–2135 (2010)
51. Majoros, W.H.: *Methods for Computational Gene Prediction*. CUP (2007)

52. Stanke, M., Schöffmann, O., Morgenstern, B., Waack, S.: Gene prediction in eukaryotes with a generalized hidden markov model that uses hints from external sources. *BMC Bioinformatics* 7, 62 (2006)
53. Petersen, T.N., Brunak, S., von Heijne, G., Nielsen, H.: SignalP 4.0: discriminating signal peptides from transmembrane regions. *Nat. Methods* 8(10), 785–786 (2011)
54. Käll, L., Krogh, A., Sonnhammer, E.L.: A combined transmembrane topology and signal peptide prediction method. *J. Mol. Biol.* 338(5), 1027–1036 (2004)
55. Emanuelsson, O., Nielsen, H., Brunak, S., von Heijne, G.: Predicting subcellular localization of proteins based on their N-terminal amino acid sequence. *J. Mol. Biol.* 300(4), 1005–1016 (2000)
56. Krogh, A., Larsson, B., von Heijne, G., Sonnhammer, E.L.: Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes. *J. Mol. Biol.* 305(3), 567–580 (2001)
57. Horton, P., Park, K.J., Obayashi, T., Fujita, N., Harada, H., Adams-Collier, C.J., Nakai, K.: WoLF PSORT: protein localization predictor. *Nucleic Acids Res.* 35, W585–W587 (2007)
58. Blum, T., Briesemeister, S., Kohlbacher, O.: MultiLoc2: integrating phylogeny and gene ontology terms improves subcellular protein localization prediction. *BMC Bioinformatics* 10, 274 (2009)
59. Li, L., Stoeckert Jr., C.J., Roos, D.S.: OrthoMCL: identification of ortholog groups for eukaryotic genomes. *Genome Res.* 13(9), 2178–2189 (2003)
60. Ostlund, G., Schmitt, T., Forslund, K., Köstler, T., Messina, D.N., Roopra, S., Frings, O., Sonnhammer, E.L.: InParanoid 7: new algorithms and tools for eukaryotic orthology analysis. *Nucleic Acids Res.* 38, D196–D203 (2010)
61. Altenhoff, A.M., Schneider, A., Gonnet, G.H., Dessimoz, C.: OMA 2011: orthology inference among 1000 complete genomes. *Nucleic Acids Res.* 39, D289–D294 (2011)
62. Kurtz, S., Phillippy, A., Delcher, A.L., Smoot, M., Shumway, M., Antonescu, C., Salzberg, S.L.: Versatile and open software for comparing large genomes. *Genome Biol.* 5(2), R12 (2004)
63. Soderlund, C., Nelson, W., Shoemaker, A., Paterson, A.: SyMAP: A system for discovering and viewing syntenic regions of fpc maps. *Genome Res* 16(9), 1159–1168 (2006)
64. Green, M.L., Karp, P.D.: Using genome-context data to identify specific types of functional associations in pathway/genome databases. *Bioinformatics* 23(13), i205–i211 (2007)
65. Notebaart, R.A., van Enckevort, F.H., Francke, C., Siezen, R.J., Teusink, B.: Accelerating the reconstruction of genome-scale metabolic networks. *BMC Bioinformatics* 7, 296 (2006)
66. Plata, G., Fuhrer, T., Hsiao, T.L., Sauer, U., Vitkup, D.: Global probabilistic annotation of metabolic networks enables enzyme discovery. *Nat. Chem. Biol.* (September 9, 2012)
67. Murphy, C., Wu, M., Butler, G., Tsang, A.: Curation of characterized glycoside hydrolases of fungal origin. *Database* (May 26, 2011)
68. Cvijovic, M., Olivares-Hernández, R., Agren, R., Dahr, N., Vongsangnak, W., Nookaew, I., Patil, K.R., Nielsen, J.: BioMet toolbox: genome-wide analysis of metabolism. *Nucleic Acids Res.* 38, W144–W149 (2010)
69. Brown, D.P., Krishnamurthy, N., Sjölander, K.: Automated protein subfamily identification and classification. *PLoS Comput. Biol.* 3(8), e160 (2007)
70. Plewniak, F., Bianchetti, L., Brelivet, Y., Carles, A., Chalmel, F., Lecompte, O., Mochel, T., Moulinier, L., Muller, A., Muller, J., Prigent, V., Ripp, R., Thierr, J.C., Thompson, D.T., Wicker, N., Poch, O.: PipeAlign: A new toolkit for protein family analysis. *Nucleic Acids Res.* 31(13), 3829–3832 (2003)

# Mining Anti-coagulant Drug-Drug Interactions from Electronic Health Records Using Linked Data

Jyotishman Pathak, Richard C. Kiefer, and Christopher G. Chute

Division of Biomedical Statistics and Informatics, Department of Health Sciences Research,  
Mayo Clinic College of Medicine, Rochester, MN 55905, USA  
{Pathak.Jyotishman, Kiefer.Richard, Chute}@mayo.edu

**Abstract.** By nature, healthcare data is highly complex and voluminous. While on one hand, it provides unprecedented opportunities to identify hidden and unknown relationships between patients and treatment outcomes, or drugs and allergic reactions for given individuals, representing and querying large network datasets poses significant technical challenges. In this research, we study the use of Semantic Web and Linked Data technologies for identifying potential drug-drug interaction (PDDI) information from publicly available resources, and determining if such interactions were observed using real patient data. Specifically, we apply Linked Data principles and technologies for representing patient data from electronic health records (EHRs) at Mayo Clinic as Resource Description Framework (RDF), and identify PDDIs for widely prescribed anti-coagulant Warfarin. Our results from the proof-of-concept study demonstrate the potential of applying such a methodology to study prescription trends based on gender and age as well as patient health outcomes.

**Keywords:** Drug-drug interactions, SPARQL, Federated querying, DrugBank, Electronic Health Records.

## 1 Introduction

An important aspect in realizing the vision of translational research lies in the ability to access, integrate, analyze and manage multiple and heterogeneous datasets within and across functional domains. This necessitates a systematic study of clinical phenotypes and health-related treatment outcomes to better understand the impact of effective patient-care management. The Semantic Web, and its related tools and technologies provide the underlying infrastructure for large-scale data integration and knowledge acquisition, and are being increasingly adopted by the biological, clinical and translational science communities to serve their information management and querying requirements. In particular, the Linked Open Data (LOD [1]) community project at the World Wide Web Consortium (W3C) is publishing various open data sets as Resource Description Framework (RDF [2]) on the Web and extending it by setting RDF links between data items from different data sources containing information about genes, proteins, pathways, diseases, and drugs. While this presents a very powerful platform for federated querying and heterogeneous data integration,

its true potential can only be realized when combining such information with “real” patient data from electronic health records. However, in practice, due to several privacy, security, ethical, policy and confidentiality issues, patient data is closely guarded and monitored for unauthorized access within institutional firewall boundaries. Consequently, projects such as the LOD rely on “sample” patient data that do not represent the inherent idiosyncrasies and complexities of information contained within an electronic health record system.

In this manuscript, we report early experiences by leveraging our prior work [3, 4] in applying linked data principles for representing patient data from electronic health record systems at Mayo Clinic to study drug-drug interaction patterns for a widely prescribed anti-coagulant Warfarin. In particular, we use open-source tooling and standardized ontologies for creating virtual RDF graphs (i.e., “views”) from Mayo’s clinical enterprise warehouse and demonstrate federated querying for potential drug-drug interaction (DDI) information using public data available from the Linked Open Data cloud. It is well-known that adverse drug events are a major health risk, and DDIs are one of the causes of such events. However, while thousands of DDIs have been reported, only a handful is worth any attention. Furthermore, a set of DDIs that suit one medical center or patient care facility might not be entirely appropriate to others. Consequently, there is significant research and on-going debate on how DDI information can be leveraged for better care, particularly in the context of clinical decision support and EHRs [5].

The work presented in this study is informed by such advances and focuses primarily on mining DDI information from EHR systems using Semantic Web technologies. Specifically, our proof-of-concept described in this manuscript is based on DDI information for Warfarin (Brand Name: Coumadin). Our reasoning behind selecting Warfarin for demonstration of our methods was not only due to the fact it is a commonly prescribed anti-coagulant medication, but also because existing literature [6, 7] have demonstrated the role of genes to guide the drug administration and dosing recommendations—an area of future interest and relevance to our project [4]. The preliminary results from our study further illustrate the strong potential for considering Semantic Web technologies to enabling Web-scale data integration and federation in biomedical research and development.

We begin by providing a brief background to Semantic Web technologies and their benefits, followed by a description of Mayo Clinic’s clinical data repository and the DrugBank and NCBO BioPortal public knowledgebases.

## 2 Background and Materials

### 2.1 Semantic Web and Related Technologies

A key benefit of using Semantic Web technologies is a rigorous mechanism of defining and linking data using Web protocols for automation, integration and reuse across various applications. Specifically, an “attractive” element of the Semantic Web is its simple data model, called Resource Description Framework (RDF), which represents data as a labeled graph connecting resources and their attribute values with labeled edges representing properties. The graph can be structurally parsed into a set

of triples (subject, predicate, object), making it very general and easy to express any type of data. Such a model coupled with (i) dereferenceable Uniform Resource Identifiers (URI's) for creating globally unique names, and (ii) standard languages such as RDFS, OWL, and SPARQL for creating ontologies as well as modeling and querying data, provides a very powerful framework for the integration of heterogeneous data. Of particular relevance to this study is the Linked Open Data (LOD [1]) initiative from the World Wide Web Consortium (W3C) that aims to bootstrap the Web of data by publishing existing data sets in RDF on the Web and creating numerous links between them. As of February 2013, the LOD project has more than 300 public datasets from multiple domains (e.g., genes, drugs and side effects, diseases, anatomy) with approximately 300 billion triples connected via more than 500 million links, and comprises resources such as DBpedia [8]) that provide an RDF representation of Wikipedia.

## 2.2 Enterprise Clinical Data Warehouse at Mayo Clinic

Mayo Clinic has a history of over 100 years in organizing patient records to support research and quality improvements [9]. Starting with structured paper documents storing patient information about laboratory results or physical examination findings, Mayo has supported the notion of explicitly missing information since 1907. Around 1990, efforts to integrate and organize information into semantically well-formed structure were initiated in tandem with infrastructure development for registry creation and information retrieval. The Mayo Clinic Life Sciences System (MCLSS [10]) is a rich clinical data repository maintained by the Enterprise Data Warehousing Section of the Department of Information Technology. MCLSS contains patient demographics, diagnoses, hospital, laboratory, flowsheet, clinical notes and pathology data obtained from multiple clinical and hospital source systems within Mayo Clinic at Rochester, Minnesota. Data in MCLSS is accessed via the Data Discovery and Query Builder (DDQB) toolset, consisting of a web-based GUI application and programmatic API. Investigators, study staff and data retrieval specialists can utilize DDQB and MCLSS to rapidly and efficiently search millions of patient records. Users are able to quickly build, save and share complex queries without programming or database knowledge. A unique text search engine provides the capability to rapidly search for specific words and phrases within unstructured text documents, such as clinical notes and pathology reports, freeing investigators from many hours of tedious manual chart reviews. Data found by DDQB can be exported into CSV, TAB or Excel files for portability. It implements full data authorization and audit logging to ensure data security standards are met.

For this project, we leverage MCLSS to access and retrieve patient demographic and diagnosis data, and represent such knowledge using Resource Description Framework (RDF). We also leverage patient clinical information from Mayo's Electronic Medical Record and Genomics (eMERGE [11]) project comprising approximately 7000 patients. In particular, we represent the medication prescription data from this cohort as RDF graphs for querying. We discuss additional details and processes involved in the remainder of this manuscript.



### 2.3 DrugBank and Drug-Drug Interaction Information

DrugBank [12] is a publicly available rich source of annotated drugs and drug target information. At the time of writing this manuscript in February 2013, DrugBank contained 6711 drug entries including 1447 FDA-approved small molecule drugs, 131 FDA-approved biotech (protein/peptide) drugs, 85 nutraceuticals and 5080 experimental drugs. Additionally, 4227 non-redundant protein (e.g., drug transporter or carrier) sequences are linked to these drug entries. The data is represented via a DrugCard where each entry comprising 150 data fields contains information on drug/chemical and drug target or protein data. Specifically, the data fields include information drug-action pathways, drug transporter data, drug metabolite data, pharmacogenomic data, adverse drug response data, ADMET data, pharmacokinetic data, computed property data and chemical classification data, and more recently drug-drug and drug-food interactions. Additionally, the DrugBank data is also available as RDF via the Bio2RDF [13] SPARQL endpoint.

Our objective is to use the SPARQL endpoint to demonstrate how we can integrate publicly available data with institutional EHR data in a flexible manner. Since our use case for this work is investigate DDI information for Warfarin (Coumadin), we restrict our SPARQL query searches to these entities (see more details in Section 3.2). Table 1 below illustrates a partial listing of DDI information extracted from DrugBank for Warfarin and how the combination of the two effects the INR (international normalized ratio) levels.

**Table 1.** Partial listing of Warfarin drug-drug interactions

<b>May increase INR (i.e., a lower dose of Warfarin may be necessary)</b>	<b>May decrease INR (i.e., a higher dose of Warfarin may be necessary)</b>
Amiodarone	Antacids
Antifungals (Fluconazole)	Anticonvulsants (Phenytoin)
Antiretrovirals (Delaviradine)	Barbiturates
Cimetidine	Bile acid resins
Corticosteroids	Cyclosporine
Fibrates	Rifampin
Griseofulvin	
Isoniazid	
Leflunomide (Arava)	
Mifepristone	
Orlistat (Alli, Xenical)	
Protein pump inhibitors (Omeprazole)	
Statins (Simvastatin)	
Tamoxifen	
Thyroid replacement	
Tramadol (Ultram)	
<b>Concomitant use may increase bleeding risk on Warfarin</b>	
Anti-depressants (Celexa, Lexapro)	
Anti-inflammatories (Celecoxib)	
Anti-plateletss (Clopidogrel)	
Anti-coagulants (Heparin)	

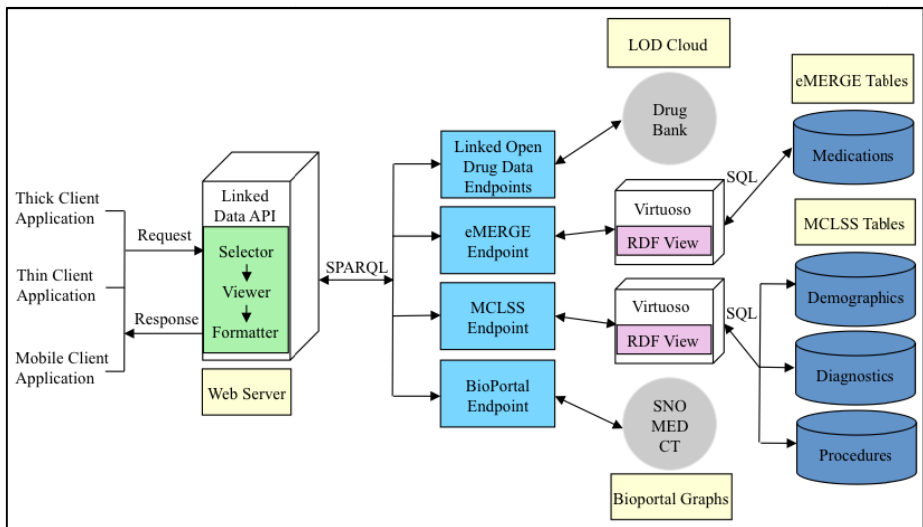
## 2.4 National Center for Biomedical Ontology (NCBO) BioPortal

The National Center for Biomedical Computing (NCBO) provides a national computing infrastructure—BioPortal [14]—to access a library of biomedical ontologies and terminologies. BioPortal enables community participation in the evaluation and evolution of ontologies by providing features to add mappings between terms, to add comments linked to specific ontology terms and to provide ontology reviews. In this work, we leverage BioPortal's SPARQL endpoint to query multiple ontologies.

## 3 Methods

### 3.1 Representing Patient Clinical Data as RDF Graphs

Figure 1 shows the proposed architecture for representing patient health records from MCLSS using RDF, linked data and related technologies. It comprises of two main components: (1) data access and storage, and (2) SPARQL-based querying interface. Here we provide a brief overview of these components; more details are described in prior work [3].



**Fig. 1.** Linked Clinical Data Architecture

*Data Access and Storage.* This component comprises the patient demographics, diagnoses, procedures, laboratory results, and free-text clinical and pathology notes generated during a clinical encounter. Since our goal is to represent the data stored in the MCLSS database as RDF data, we use the Virtuoso Universal Server [15] that acts as a mediator in the creation of materialized RDF graphs as well as provides a SPARQL endpoint for querying the graphs. In particular, R2RML is used to describe

the mappings between the relational schema and RDFS/OWL ontologies to create the RDF triples. This language generates a mapping file from table structures of the databases in eMERGE and MCLSS that can then be customized by replacing the auto-generated terms with concepts from standardized ontologies. In our case, we modified the custom ontology generated by Virtuoso for creating these mappings with terms and concepts from the SNOMED-CT [15] standardized ontology.

*SPARQL endpoint.* The RDF graphs created from MCLSS and eMERGE using the above approach was exposed via two different endpoints. This allows software application clients to query the MCLSS RDF data using the SPARQL query language. Given that our overarching goal is to integrate the eMERGE, MCLSS and DrugBank RDF graphs, our objective is to execute federated queries across multiple SPARQL endpoints. The MCLSS and eMERGE endpoints were placed on two different servers in order to reduce the bandwidth load on the machines during the query execution. We discuss the details of SPARQL-based federated querying in the next section.

### 3.2 SPARQL-Based Federated Querying for Drug-Drug Interactions

As shown in Figure 1, our goal is to federate between three main data sources: eMERGE, MCLSS and DrugBank, where eMERGE is a MySQL database containing prescription data, MCLSS is a DB2 database containing patient clinical and demographic data, and the DrugBank is a public drug data repository. Since our interest lies in querying for DDI pairs in DrugBank and determining such potential interactions using Mayo's EMR data, in its current form, one would have to execute multiple SQL query across all these datasources to retrieve the appropriate resultset. Instead, by leveraging RDF and Semantic Web technologies, we demonstrate how this can be achieved using a single SPARQL query.

In particular, there were four different endpoints queried during this study. The first endpoint was DrugBank where we state the drug for which is the basis for our investigation into patient prescription histories. By stating Warfarin in the first service statement we are able to retrieve the list of drugs which have been established in DrugBank to have a DDI. The next service statement uses that list of medications to find the patients who have been prescribed both Warfarin and a DDI medication on the same clinic visit. The form of the results from this portion of the query is in patient clinic numbers. A decision was made at this point of the query that the list of patients would be distinct so those who were prescribed a specific DDI combination with Warfarin would not be duplicated in the list.

When creating the mapping file for the RDF view, SNOMEDCT concept code 422549004 was used to represent patient clinic numbers in order to align our database concepts with accepted ontology descriptions. In this case, the concept id represents a "patient related identification code". For the same reasoning, SNOMEDCT concept code 432213005 was mapped to the clinic visit dates in the database as they represent the "date of diagnosis". Medications listed in the table were mapped to SNOMEDCT concept code 33633005 as we considered them a "medication prescription".

```

PREFIX drug:<http://bmiddev4:8890/drugbank#>
PREFIX emerge:<http://hsrdev02:8890/emerge/>
PREFIX demo:<http://edison:8890/demo/>
PREFIX diag:<http://edison:8890/diag/>
PREFIX skos:<http://www.w3.org/2004/02/skos/core#>
PREFIX snomed:<http://purl.bioontology.org/ontology/SNOMEDCT#>

SELECT ?ddiDrug1 ?diag ?label (COUNT(?diag) AS ?diagCount)
WHERE {
  SERVICE<http://bmiddev4:8890/sparql> {
    ?s1 drug:name "Warfarin" .
    ?s1 drug:drug-interactions_drug-interaction_name ?ddiDrug1 .
  }
  {
    SELECT distinct ?drug1 ?patientId2
    WHERE {
      SERVICE<http://hsrdev02:8890/sparql> {
        ?s2 snomed:33633005 "Warfarin" .
        ?s2 snomed:422549004 ?patientId2 .
        ?s2 snomed:432213005 ?date2 .
        ?s3 snomed:33633005 ?ddiDrug1 .
        ?s3 snomed:422549004 ?patientId3 .
        ?s3 snomed:432213005 ?date3 .
        filter(?patientId2=?patientId3)
        filter(?date2=?date3)
      }
    }
  }
  SERVICE<http://edison.mayo.edu:8890/sparql> {
    ?s4 snomed:422549004 ?patientId2 .
    ?s4 snomed:396278008 ?internalId4 .
    ?s5 snomed:396278008 ?internalId5 .
    ?s5 snomed:8319008 ?diag
    filter(?internalId4=?internalId5)
  }
  SERVICE <http://sparql.bioontology.org/sparql?apikey=24e0350e-54e0-11e0-9d7b-005056aa3316> {
    GRAPH <http://bioportal.bioontology.org/ontologies/ICD9CM> {
      ?s skos:prefLabel ?label .
      ?s skos:notation ?diag .
    }
  }
}
GROUP BY ?drug1 ?diag
ORDER BY ?drug1 DESC(?diagCount)

```

**Fig. 2.** SPARQL query to retrieve patient diagnosis information

The next service statement queries the MCLSS endpoint against two different graphs. The demographic graph was required to transform the list of patient clinic numbers into internal MCLSS key ids as they are used as primary keys on all the tables in the database. The query then matches the internal primary key from the demographic graph with the internal primary key in the diagnostic graph. Once that match is made, we are able to extract the list of diagnoses made for the patients on the date of the clinic visit.

When mapping the MCLSS tables, SNOMEDCT concept code 422549004 was also used to map the patient clinic identification numbers. Another advantage to using known ontologies while doing the mappings is to provide a common understanding of what the values within each table represent. If we were to use column names, the patient numbers in eMERGE would use the predicate “clinic” while the ones in MCLSS demographics table would use the predicate “mcn”. To differentiate patient clinic numbers with internal table identification numbers, we used SNOMEDCT concept code 396278008 to map the primary table keys to the ontology’s “identification number” representation. The relationship between patient and the diagnosis was mapped as SNOMEDCT concept code 8319008 to represent the “principle diagnosis”.

The list of diagnoses from the MCLSS diagnostic graph is in the form of ICD9CM codes. Rather than manually matching up codes with their descriptions, we end the query by using a service statement against the ICD9CM graph located at the BioPortal endpoint. This endpoint was added to the initial three to help us match the list of diagnosis code values with the code descriptions to include in the final result set.

The results of the query are grouped by medication and diagnosis. We used a count function on the diagnosis values in order to analyze the results outlined in the next section. A graphical representation of the query process is shown in Figure 3.

Two other queries were run to further the analysis of patient data. One query provided a breakdown of the ages of the patients when they were prescribed the DDI combination and the other broke it down in terms of gender. Patient age information is included as a part of eMERGE, and hence service calls to MCLSS and DrugBank were not required. Similarly, gender information is included in MCLSS, and hence the diagnosis graph matching was not required nor the DrugBank service call.

In the interest of space in this paper, the queries have been posted to our project wiki: [http://informatics.mayo.edu/LCD/index.php/Project\\_Warfarin\\_DDI](http://informatics.mayo.edu/LCD/index.php/Project_Warfarin_DDI).

## 4 Results

We retrieved DDI information from the prescription medication data using the SPARQL endpoints for Warfarin (Coumadin) on a cohort of 6758 patients that are participating in the Mayo Clinic eMERGE (Electronic Medical Records and Genomics) study [11]. This cohort primarily comprises of elderly patients who have been diagnosed with cardiovascular diseases; a detailed description of the cohort is presented elsewhere [11]. Figures 4 and 5 show the gender and age distributions, respectively, on the eMERGE cohort for which potential DDIs were observed using the electronic health record data retrieved from the MCLSS SPARQL endpoint. Figure 4 highlights the gender distribution and as such, each distinct patient was included only once in the result set data. Figure 5 highlights the age distribution and

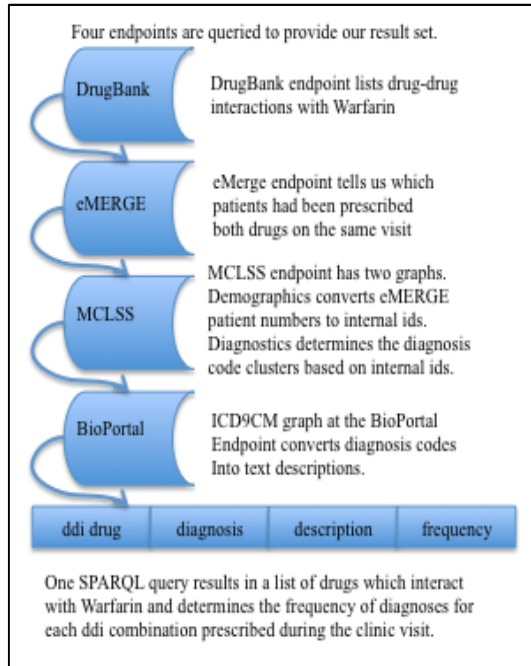
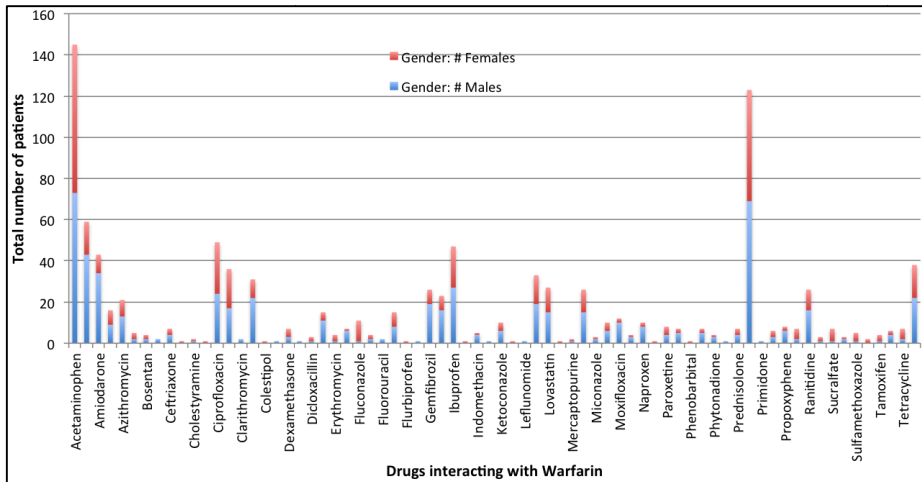


Fig. 3. SPARQL query process

because a patient may have had the same ddi prescribed more than once in their lifetime, each incidence was included. Of the total 171 potential DDIs for Warfarin (Table 1 shows only a small subset), 72 DDIs were observed in the prescription data for at least one or more patient. While the gender distribution was not biased, the potential DDI evidences were observed more primarily in the elderly (65 years or above). This would be consistent with the fact that cardiovascular related complications are more prevalent in the older population, and hence anticoagulants are prescribed to prevent thrombosis and thromboembolism.



**Fig. 4.** Gender distribution for the eMERGE cohort (N=6758) for potential Warfarin DDIs

Specifically, the most commonly prescribed medications that interact with Warfarin were Acetaminophen and Prednisone. The distribution among gender was relatively equal. Prednisone was the most commonly prescribed medication among patients in the 18-30 year old grouping. Dicloxacillin was given to age groups 18-30 and 71+ but not to 31-50 or 51-70. Males were almost 3 times more likely to be prescribed Allopurinol and 4 times more likely to be prescribed Amiodarone. Females were 10 times more likely to be prescribed Fluconazole. Older patients were more likely to take Clopidogrel – as usually prescribed to patients receiving stents. Patients in the age group of 71 years or above were only given Phytonadione—a drug used to reverse the effects of Warfarin when INR (international normalized ratio) levels were too low.

Finally, Table 2 shows another dimension of these findings in the context of patient disease diagnoses (based on ICD-9-CM), which further illustrates a higher prevalence of cardiovascular diseases and its risk factors in this cohort. Specifically, we observe a large number of patients diagnosed with Essential Hypertension (ICD-9 code 401.9), followed by Diabetes Mellitus (ICD-9 code 250) and Hyperlipidemia (ICD-9 code 272.4). This would correlate with the fact most of the patients were prescribed

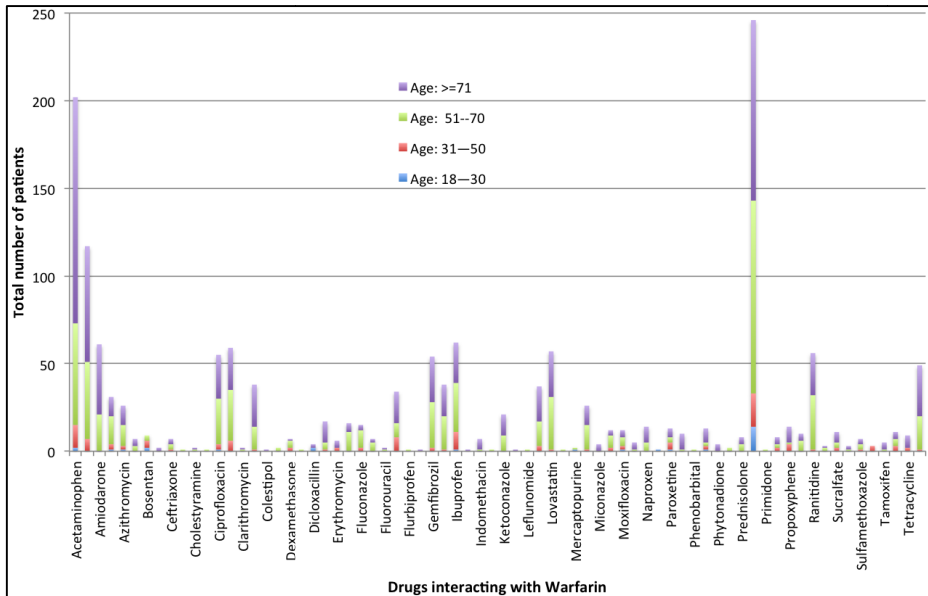


Fig. 5. Age distribution for the eMERGE cohort (N=6758) for potential Warfarin DDIs

cardiovascular drugs including statins and antiplatelets. Further, with this particular cohort primarily comprising of elderly with a mean age of 60.3 years and several comorbidities, we also observe higher rates of prescriptions for antibiotics and antivirals.

## 5 Summary

### 5.1 Discussion

The overarching goal of this study is to precisely explore federated data integration and querying using public data sources from the Linked Open Data cloud, and private, identifiable patient data from Mayo Clinic's EHR systems. Using open-source tooling and software, we developed a proof-of-concept system that allows representing patient data stored in Mayo's enterprise warehouse system as RDF, and exposing it via a SPARQL end-point for accessing and querying. Our use case for federated querying of DDI information from DrugBank and the NCBO BioPortal SPARQL endpoints further demonstrated the applicability of such a system and the benefits of interlinking and querying multiple, heterogeneous Web data sources that are publicly available, with private (and institution-specific) patient information. We hypothesize that further development of such a system can immensely facilitate, and potentially accelerate scientific findings in clinical and translational research.

We also acknowledge that several potential DDIs relevant to Warfarin are often known to the clinical care providers, and are in fact prescribed in combinations,

although with appropriate dosing considerations. For example, we observed that 37 out of 38 patients prescribed Clopidogrel and Warfarin concomitantly were 50 years old or above. This would make clinical "sense" since individuals in this age group frequently receive drug-eluting stents (Clopidogrel prevents blood clotting). Similarly, concomitant prescription for Ciprofloxacin was observed in 51 out of 55 patients that were above 50 years since it is an important antibiotic used widely to treat several respiratory, urinary tract, gastrointestinal and abdominal infections. Consequently, while concurrent administration cannot be avoided entirely, and hence may increase or decrease Warfarin activity, the clinical protocols and guidelines recommend close monitoring of daily INR levels, and appropriately adjusting Warfarin dosing. For instance, Isoniazid—a commonly used drug to prevent and treat tuberculosis—when co-administered with Warfarin, often increases the anticoagulant effects of Warfarin by interfering with the enzyme in the liver that eliminates Warfarin, thereby necessitating dose adjustments (lowering the Warfarin dose in this case).

## 5.2 Limitations

The proof-of-concept system developed in this study has several limitations. First, while we demonstrated the applicability of the system via sample use case queries, a more robust and rigorous evaluation along several dimensions (e.g., performance, query response, precision and recall of query results etc.) is required before it can be deployed within an enterprise environment. Note that since our use cases are based on federated querying of several public SPARQL endpoints, the system performance and query responses are dependent on the behavior of the endpoints. Nevertheless, we plan to perform a thorough system evaluation after the integration of additional MCLSS sources (e.g., laboratory, clinical and pathology reports) that contain large amounts of patient data. Second, the use-case queries were executed on a small cohort of approximately 7000 patients, and only drug prescription data was available. It remains to be seen if the preliminary findings for the DDI pairs can be replicated in a larger cohort, and more importantly, using drug administration data. Finally, while one of the Linked Data principles is to make data publicly available and accessible, due to privacy and HIPPA constraints of identifiable patient data, the MCLSS RDF views remain private. Consequently, only appropriate personnel within Mayo's firewall approved by Mayo's Institutional Review Board participating in this study can access our application.

## 5.3 Future Work

In addition to addressing the limitations aforementioned, there are several activities that we plan to pursue in the future. Firstly, in this study, we studied only a handful of DDI pairs. Our immediate goal is to expand the DDI pair list that are of clinical significance and consider both drug prescription and administration data. Secondly, our experience in discussing and demoing the proof-of-concept to clinicians made it amply clear that we should focus on developing visual and interactive interfaces for



**Table 2.** Top 5 ICD-9-CM diagnosis (for the cohort in Figures 4 and 5)

Drug interacting with Warfarin	ICD-9-CM diagnosis codes	Top 5 observed ICD-9-CM diagnosis
Acetaminophen	401.9	Unspecified essential hypertension
	V58.61	Encounter for long-term (current) use of anticoagulants
	250	Diabetes mellitus without mention of complication
	272.4	Other and unspecified hyperlipidemia
	414	Coronary atherosclerosis of nonautologous biological bypass graft
Clopidogrel	414	Coronary atherosclerosis of nonautologous biological bypass graft
	401.9	Unspecified essential hypertension
	250	Diabetes mellitus without mention of complication
	272.4	Other and unspecified hyperlipidemia
	414.01	Coronary atherosclerosis of native coronary artery
Doxycycline	401.9	Unspecified essential hypertension
	250	Diabetes mellitus without mention of complication
	414	Coronary atherosclerosis of nonautologous biological bypass graft
	272.4	Other and unspecified hyperlipidemia
	714	Felty's syndrome
Gemfibrozil	250	Diabetes mellitus without mention of complication
	V58.61	Encounter for long-term (current) use of anticoagulants
	V42.0	Kidney replaced by transplant
	401.9	Unspecified essential hypertension
	272.4	Other and unspecified hyperlipidemia
Hydrocortisone	401.9	Unspecified essential hypertension
	250	Diabetes mellitus without mention of complication
	414	Coronary atherosclerosis of nonautologous biological bypass graft
	272.4	Other and unspecified hyperlipidemia
	714	Felty's syndrome
Ibuprofen	401.9	Unspecified essential hypertension
	250.01	Type 1 diabetes mellitus
	V58.61	Encounter for long-term (current) use of anticoagulants
	427.31	Atrial fibrillation
	272.4	Other and unspecified hyperlipidemia
Lovastatin	250	Diabetes mellitus without mention of complication
	401.9	Unspecified essential hypertension
	272.4	Other and unspecified hyperlipidemia
	V58.61	Encounter for long-term (current) use of anticoagulants
	414	Coronary atherosclerosis of nonautologous biological bypass graft
Prednisone	401.9	Unspecified essential hypertension
	V42.0	Kidney replaced by transplant
	V58.61	Encounter for long-term (current) use of anticoagulants
	250	Diabetes mellitus without mention of complication
	V42.1	Heart replaced by transplant
Triamcinolone	401.9	Unspecified essential hypertension
	V58.61	Encounter for long-term (current) use of anticoagulants
	427.31	Atrial fibrillation
	272.4	Other and unspecified hyperlipidemia
	414	Coronary atherosclerosis of nonautologous biological bypass graft

forming the SPARQL queries. To this end, we plan to explore visual SPARQL editing tools, such as SPARQLMotion [16]. Finally, as depicted in Figure 1, we plan to use the Linked Data API for creating our service layer to provide application developers a friendlier access to the data, for example, using JSON.

**Acknowledgment.** This research is supported in part by the Mayo Clinic Early Career Development Award (FP00058504) and the eMERGE consortia (U01-HG-006379). The authors would also like to thank Jamie Kiefer for her inputs and feedback on the manuscript.

## References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems* 5(3), 1–22 (2009)
2. Resource Description Framework (RDF) (2011), <http://www.w3.org/RDF/> (cited January 13, 2011)
3. Pathak, J., Kiefer, R., Chute, C.: Applying Linked Data Principles to Represent Patient’s Electronic Health Records at Mayo Clinic: A Case Report. In: 2nd ACM SIGHT International Health Informatics Symposium 2012 (2012)
4. Pathak, J., Kiefer, R., Chute, C.G.: Using Semantic Web Technologies for Cohort Identification from Electronic Health Records to Conduct Genomic Studies. *AMIA Summit on Clinical Research Informatics (CRI) American Medical Informatics Association (AMIA)*, p. 10–19 (2012)
5. Greenberg, M., Ridgely, M.: CLinical decision support and malpractice risk. *JAMA: The Journal of the American Medical Association* 306(1), 90–91 (2011)
6. Ramirez, A.H., et al.: Predicting warfarin dosage in European-Americans and African-Americans using DNA samples linked to an electronic health record. *Pharmacogenomics* 13(4), 407–418 (2012)
7. Delaney, J.T., et al.: Predicting Clopidogrel Response Using DNA Samples Linked to an Electronic Health Record. *Clin. Pharmacol. Ther.* 91(2), 257–263 (2012)
8. Bizer, C., et al.: DBpedia - A crystallization point for the Web of Data. *Web Semant.* 7(3), 154–165 (2009)
9. Kurland, L., Molgaard, C.: The Patient Record in Epidemiology. *Scientific American* 245(4), 54–63 (1981)
10. Weiss, T.: IBM, Mayo Clinic to develop database for clinical trials, research (2002), [http://www.computerworld.com/s/article/69540/IBM\\_Mayo\\_Clinic\\_to\\_develop\\_database\\_for\\_clinical\\_trials\\_research](http://www.computerworld.com/s/article/69540/IBM_Mayo_Clinic_to_develop_database_for_clinical_trials_research) (June 8, 2011)
11. Kullo, I., et al.: Leveraging Informatics for Genetic Studies: Use of the Electronic Medical Record to Enable a Genome-Wide Association Study of Peripheral Arterial Disease. *JAMIA* 17(5), 568–874 (2010)
12. Knox, C., et al.: DrugBank 3.0: a comprehensive resource for ‘Omics’ research on drugs. *Nucleic Acids Research* 39(suppl. 1), D1035–D1041 (2011)
13. Belleau, F., et al.: Bio2RDF: Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics* 41(5), 706–716 (2008)
14. Musen, M.A., et al.: The National Center for Biomedical Ontology. *Journal of the American Medical Informatics Association* 19(2), 190–195 (2012)
15. Virtuoso Universal Server (2011), <http://virtuoso.openlinksw.com/> (cited January 16, 2011)
16. Waldman, S.: TopQuadrant: SPARQLMotion Visual Scripting Language (June 28, 2011), <http://www.topquadrant.com/products/SPARQLMotion.html>

# Author Index

- Asiaee, Amir H. 38
- Baker, Christopher J.O. 69
- Booth, David 54
- Bot, Jan J. 23
- Bukhari, Ahmad C. 69
- Butler, Greg 113
- Chute, Christopher G. 128
- de Vries, Arjen P. 23
- Doshi, Prashant 38
- Dos Reis, Julio Cesar 90
- Gombocz, Erich Alfred 1
- Groß, Anika 81, 90
- Hartung, Michael 81, 90
- Hulsman, Marc 23
- Kiefer, Richard C. 128
- Klein, Artjom 69
- Kolb, Lars 81
- Koschmieder, André 46
- Krauthammer, Michael 105
- Lebo, Timothy 105
- Leser, Ulf 46
- McCusker, Jamie P. 105
- McGuinness, Deborah L. 105
- Minning, Todd 38
- Parikh, Priti 38
- Pathak, Jyotishman 128
- Pruski, Cédric 90
- Rahm, Erhard 81, 90
- Reinders, Marcel J.T. 23
- Sahoo, Satya 38
- Sheth, Amit 38
- Stoltmann, Thomas 46
- Tarleton, Rick L. 38
- Zimmermann, Karin 46