Querying Provenance along with External Domain Data Using Prolog

Wellington Oliveira[1,2], Kary A. C. S. Ocaña[3], Daniel de Oliveira[1], and Vanessa Braganholo[1]

[1] Universidade Federal Fluminense, Brazil
{wellmor, danielcmo, vanessa}@ic.uff.br
[2] Instituto Federal do Sudeste de Minas Gerais - Rio Pomba Campus, Brazil
[3] Laboratório Nacional de Computação Científica, Brazil
karyann@lncc.br

Abstract. Bioinformaticians have relied on computational simulations to run their biological experiments. This is due to the advantages offered by existing approaches, including tools to manage and run experiments, verify results and capture/analyze provenance data. Provenance is metadata that helps scientists to analyze *in silico* experiments, better understand their results, and reproduce them. However, provenance data is usually not enough. To improve the knowledge about the experiment, scientists often need to use domain-specific data available on external sources along with provenance data that is captured during the experiment execution. Although most of the existing tools provide mechanisms to capture and analyze provenance data, they do not offer means to enrich provenance with external domain data, or, when they do it, they do not have mechanisms to query provenance and domain data together in an effective way. In this article, we present an approach to analyze provenance and domain data together using Prolog. Our goal is to improve provenance analysis. As a proof of concept, we present a case study of phylogenetic analysis (a biological experiment). Our approach, however, is designed to be generic and can be applied to other domains.

Categories and Subject Descriptors: H.2 [**Database Management**]: Miscellaneous

Keywords: provenance analysis, scientific experiments, workflows

1. INTRODUCTION

In the past decades, several categories of bioinformatic experiments (*e.g.*, phylogeny, phylogenomics, comparative genomics, metagenomics, transcriptomics, proteomics, *etc.*) became more dependent of computational simulations [Futuyma 2013]. Such simulations are usually executed countless times, with different parameters values and input data, until the biological research hypothesis can be proved or refuted. Many of these computer-based biological experiments are composed of a set of applications chained in a coherent flow, thus being suitable to be represented as scientific workflows [Mattoso et al. 2010]. The paradigm of scientific workflows is an abstraction that models a set of activities (application executions) and the data dependencies among them. Each workflow is part of a scientific experiment and follows the steps of the scientific experiment life cycle defined by Mattoso et al. [2010]. A fundamental and critical phase of the scientific experiment's life cycle is the analysis, when scientists look for answers to their research hypothesis in the experiment's results [Mattoso et al. 2010].

Scientific workflows are usually modeled and executed by engines called Scientific Workflow Management Systems (*i.e.* SWfMS). These systems are responsible for executing a chain of programs, encapsulated as workflow activities, and capturing important provenance data [Freire et al. 2008]. Provenance, which can be defined as an audit trail of the experiment, captures information about

Authors would like to thank CAPES, FAPERJ and CNPq for partially sponsoring the research presented in this article.

the steps (*i.e.* activities) used to produce a data product (*e.g.* data files). Using provenance data, scientists are able to analyze the quality and authorship of data and reproduce the achieved results.

The traditional definition of provenance data for scientific experiments only considers prospective provenance (*i.e.* the structure of the workflow) and retrospective provenance (*e.g.* the hour and day an activity executed, which files were produced, *etc.*). However, this type of information is often not enough for scientists to draw their conclusions, as it only registers the ownership of the data and data usage, without considering domain specific information and terminology. Let us illustrate the problem with an example derived from the evolutionary phylogenetic domain. The inference of phylogenies has many important applications in biological, technological and medical research, such as conservation in biology, biomarkers [Abu-Asab et al. 2011], and drug discovery [Searls 2003]. A phylogenetic analysis aims at generating phylogenetic trees as well as other several statistical calculations that are used to infer the evolutionary ancestry/relationships of a set of genes, species, or other taxa [Doolittle 1999]. It is considered as a computational and data intensive experiment since a huge amount of heterogeneous/unstructured data is processed by a flow of several complex bioinformatics applications such as MAFFT [Katoh and Standley 2013], ModelGenerator [Keane et al. 2006] and RAxML [Stamatakis 2014]. Common queries in this kind of experiment are: "Which workflow execution have produced the most consistent phylogenetic trees, *i.e.* with bootstrap value higher than 80?"; "Which trees contain the taxon *Bos taurus* and which parameters were used to generate those trees?". While the first query can be answered by looking inside the input files (so approaches that are able to capture this kind of provenance would be able to answer it [Gonçalves et al. 2012; Oliveira et al. 2015]), the second query can only be answered by using external data that is not part of the workflow and thus is not stored in the provenance database. In fact, it requires taking sequence IDs from the FASTA files used as input of the workflow, and retrieving associated species data from an external database such as EBI or NCBI. Thus, it cannot be answered using only the provenance captured by the SWfMS.

Answering this type of question requires bringing external domain data into the experiment repository so that the scientist can pose queries that cross knowledge from various data sources. Note that these data sources are not used in the experiment execution phase, but only in the analysis phase, and thus they are not part of the provenance that is captured at runtime. Unfortunately, related work does not solve this problem. Some approaches augment scientists' analysis capabilities by extracting domain data from input files of scientific workflows and query them through SQL [Gonçalves et al. 2012]. Despite calling this *domain data*, it is data that is already involved in the experiment execution – not data from external datasets that could complement the analysis. On the other hand, workflow systems like Taverna [Oinn et al. 2004] do allow for scientists to capture provenance and import domain data from external sources (i.e.: EBI, KEGG, etc.) by wrapping an annotation pipeline in the workflow. This, however, has three main disadvantages: (i) it is intrusive since it requires the scientist to annotate the workflow; (ii) it requires the scientist to know in advance what kind of external data she/he will need in the analysis phase of the experiment. In other words, if the experiment is run without the annotation, and only in the analysis phase the scientist realizes she/he needs external data, there is no way to include it after the experiment execution. And (iii), it provides no means of querying provenance and domain data together. Finally, related work that use ontologies to enrich provenance data suffer from the same problem: they require the annotations to be performed prior to the workflow execution [Buttler et al. 2002; Fileto et al. 2003; Sahoo et al. 2009; Sahoo et al. 2011; Alper et al. 2013].

In this article, we tackle this problem by proposing an approach that allows for scientists to import external domain data into the provenance database and to analyze provenance and domain data together. This poses several challenges: (i) What should be the granularity of the imported data? Should we import the entire external database or just part of it? It we decide to import part of it, how do we define which part should be imported?; (ii) How should we deal with the different forms of data representation? Provenance databases may be represented as relational databases or graph databases, and external data may be represented as OWL ontologies, relational databases, among other formats.

Representing these data in a single format and storing them together makes the querying process straightforward. In fact, some of these challenges have been previously recognized in the literature [Buttler et al. 2002]. To address these challenges, we develop a mechanism to import domain data from external sources and convert it to Prolog facts. Provenance data is also converted to facts and both information is stored together, thus creating a knowledge base. By translating provenance and domain data into Prolog facts, users may query them in an integrated way, and thus obtain detailed information about the experiment and improve their knowledge about it. To define the granularity, the scientist can specify a filter. Only data that satisfies the filter is imported into the knowledge base. We evaluated the proposed approach using a phylogenetic experiment (SciPhy [Ocaña et al. 2011] workflow as described in Section 4) as a case study. Using the proposed approach, we were able to answer queries that could not be answered using only provenance data.

The remainder of this article is organized as follows. Section 2 provides background on provenance data while Section 3 discusses related work. Section 4 describes the proposed approach. A case study in Section 5 assesses the advantages of our approach. Finally, Section 6 concludes and discusses future work.

2. BACKGROUND

The term "data provenance" can be defined as the source or lineage of the data, and its original usage was to interpret and reproduce the results of scientific experiments [Freire et al. 2008]. However, in recent years, the usage of provenance data has extrapolated for other goals such as scheduling [Oliveira et al. 2012] and fault tolerance [Costa et al. 2012]. In the context of scientific experiments modeled as scientific workflows, provenance can be classified as prospective and retrospective [Zhao et al. 2006]. Prospective provenance represents the specification of computational tasks. It corresponds to the steps to be followed to achieve a (final or intermediate) result. Retrospective provenance consists in a structured and detailed history of the execution of computational tasks (metadata associated with the execution of tasks and environment characteristics) [Freire et al. 2008]. It can also be specialized into explicit and implicit [Marinho et al. 2011]. Retrospective provenance is called explicit when the activity executions and the data consumed or produced by them were previously declared in the workflow specification. On the other hand, retrospective implicit provenance represents information generated by activity executions with their data accesses, modifications, and exchanges that were not explicitly declared in the workflow specification. It corresponds, for example, to the hidden provenance data captured over accessed or changed objects within implicit dataflows [Marinho et al. 2011].

There are plenty of ways to store and query provenance data. Taverna uses RDF to store provenance and annotations [Alper et al. 2013]. VisTrails [Callahan et al. 2006] stores provenance in an XML file and the log is saved into a set of MySQL tables, e-Science Central [Woodman et al. 2011] uses graph databases (*i.e.* Neo4J [Webber 2012]), while Swift/T, Pegasus, SciCumulus [Oliveira et al. 2010] and Chiron [Ogasawara et al. 2013] store provenance in relational databases. Each one of these approaches presents advantages and disadvantages. For example, relational databases rely on SQL to perform queries. Transitive closure queries cannot be implemented in these systems unless we use recursive queries, which are difficult to implement. In graph databases, transitive closure queries are easy to be performed, but queries that need to traverse more than a few graphs may not be supported. There are approaches that try to diminish this problem by summarizing the provenance graphs. Large-scale scientific experiments that are based on parameter sweeps commonly produce several small/medium-scale provenance graphs instead of a huge single provenance graph. El-Jaick et al. [2014] try to reduce the processing time of provenance queries in these scenarios by using a summary graph to answer them without the need for rebuilding the (many) original graphs. Authors showed that although they query the summarized graph, there is no data loss on query results. Although this solution works for many large provenance graphs, it does not allow for complex queries (different from transitive closure ones).

Trying to minimize the problem of diverse representation, the research community has worked on standard provenance models such as OPM [Moreau et al. 2008] and PROV [Moreau et al. 2013]. The OPM model was capable of representing only retrospective provenance. Dey et al. [2012] present a provenance model based on graphs that extends the OPM model [Moreau et al. 2008]. Instead of considering only retrospective provenance, according to the OPM model, it defines a unified provenance model that also considers prospective provenance and temporal dimension of entity relationships. Provenance queries over this model are performed using Datalog. PROV is an evolution of OPM and its data model is capable of representing transformations, ownership, *etc.* over data, along with prospective provenance. However, the provenance repositories generated by SWfMs are not always compliant with these models [Callahan et al. 2006; Oliveira et al. 2010; Woodman et al. 2011; Webber 2012; Alper et al. 2013; Ogasawara et al. 2013]. Thus, in practice, approaches that work over provenance data cannot rely on the fact that the provenance database is compliant with a given provenance model. We hope that this scenario changes in a near future.

3. RELATED WORK

To do provenance data analysis, scientists can use one of the many existing SWfMs such as Taverna [Oinn et al. 2004], Kepler [Altintas et al. 2004], Pegasus [Deelman et al. 2016], Swift/T [Wozniak et al. 2013], *etc.* These workflow systems provide mechanisms to allow scientists to pose analytical queries over provenance data, and particularly in the case of Taverna, it provides mechanisms to set Web services and bring some information from external sources. However, this needs to be made as part of the workflow. Additionally, in all cases, their provenance database is only available at the end of the workflow execution (which makes workflow steering mechanisms [Mattoso et al. 2015] harder to implement). This may be time consuming, since the workflow execution may take several days or even weeks. As an alternative, Swift/T generates a series of log files that can be analyzed to extract some useful information, but does not provide a way to query those files [Wozniak et al. 2013].

Some existing approaches [Ellqvist et al. 2009; Seltzer 2011; Zhao et al. 2008; Oliveira et al. 2016] work on the integration and querying of provenance from different sources. However, they do not integrate it with domain data. Other approaches [Gonçalves et al. 2012] do manage provenance data along with domain-specific data. However, important domain-specific data is only imported during the execution of the workflow from produced data files. Importing data from external sources during the experiment analysis is not possible.

There are also some efforts to add semantics to provenance by using ontologies [Buttler et al. 2002; Fileto et al. 2003; Sahoo et al. 2009; Sahoo et al. 2011; Alper et al. 2013]. However, they use ontologies to describe the workflow activities and/or input data, aiming at helping the scientist to better understand the experiment, its execution, and results. In this sense, these approaches are complementary to ours. However, it is far from trivial to have validated ontologies for many domains of science. Even when the ontology is validated (like many in the bioinformatics domain), some inconsistencies can be found as presented by Carvalho et al. [2011]. Also, external databases can present a huge size and most of their content may not be modeled in an ontology, thus reducing the analytic potential for scientists.

Some approaches also use ontologies in integration tasks that are performed by automated workflows [Fileto et al. 2003; Sahoo et al. 2009]. These initiatives differ from ours in the sense that we allow the scientist to use any kind of external data (not only ontologies). More importantly, we allow this to be made during and/or after the workflow execution has ended, unlike these related approaches that require annotations to be performed before the workflow execution.

In this article, we are inspired by some of the ideas of Gonçalves et al. [2012] and noWorkflow [Murta et al. 2014]. From noWorkflow, we borrowed the idea of representing the provenance database as a set of Prolog facts and rules. Note, however, that the structure of the facts and rules are

```
foo(X, Y) :- bar(X).
foo(X, Y) :- bar(Y).
foo(X, Y) :- foo2(X), foo2(Y).
```

Fig. 1. Example of rule that cannot be expressed in Datalog

completely different in both cases. In fact, in our approach, the structure of the facts depends on the structure of the provenance database the SWfMS uses – our Prolog converting tool uses this structure to automatically generate the rules and facts. noWorkflow, on the other hand, uses a fixed structure for the facts that is tight to their internal provenance representation. From Gonçalves et al. [2012], we borrow the ideas of enriching the provenance database with domain data. However, as we explained earlier in this section, their definition of domain data is limited to data within files consumed and produced during the workflow execution – they do not support external data. In the next section, we explain our approach in details.

4. ENRICHING PROVENANCE WITH DOMAIN-SPECIFIC DATA

In this article we allow scientists to enrich the provenance database with external domain data. To do so, we address two main challenges. The first one regards the data format. Since our goal is to allow scientists to query provenance and external domain data in an integrated fashion, the data must be represented in a way that makes this possible. Several data integration approaches use a canonical model [Halevy et al. 2006] and map data to this model so that the resulting mapped data can be queried. This adds an extra step that would require the help of a Computer Science (CS) specialist. Recall that different SWfMS store provenance using different representations, and external data can have a wide range of structure and semantics.

To address this challenge, in this article we translate both the provenance database and the external domain data to a knowledge base represented in Prolog. The choice of Prolog comes naturally in this scenario, since, besides being able to represent a wide variety of data [Murta et al. 2014; Oliveira et al. 2016], it also allows for more expressive power than SQL since it adds inference capabilities. Datalog would also provide these same advantages. In fact, syntactically, Datalog is a subset of Prolog, and its clauses can be parsed and executed by a Prolog interpreter [Ceri et al. 1989]. Although Datalog is more efficient on relational database queries (it processes the whole relation by using a set-oriented approach rather than the one-tuple-at-time approach of Prolog), Datalog limits the way rules can be written. As an example, it does not allow a rule to use a variable that does not appear in its body [Chong 2016]. Thus, examples such as the one on Fig. 1 would not be allowed in Datalog.

The second challenge resides in defining the granularity of the imported data. External datasets can be large, and importing them completely would waste space and query processing time. To deal with this challenge, in our approach the granularity of the imported data is defined by the scientist through a filter – only results that satisfy the filter are imported. The filter can be implemented in several different ways. It could be a SQL query, an XQuery, a SPARQL query, etc., depending on the type of the data source. In our implementation, we abstract the data source format by using web service calls that implement those queries. We then only deal with the XML answer to this query that is afterwards translated into Prolog and inserted into our knowledge base.

In the remaining of this section, we show the architecture of our proposed approach (Section 4.1), discuss our provenance knowledge base (Section 4.2), and illustrate how it could be used in a real bioinformatics experiment (Section 4.3).

4.1 Architecture

Fig. 2 shows the architecture of our proposed approach. The **SWfMS Provenance Repository** represents the database that stores provenance data generated by any SWfMS. In the context of this article, without loss of generality, we assume the SWfMS uses a relational database to store

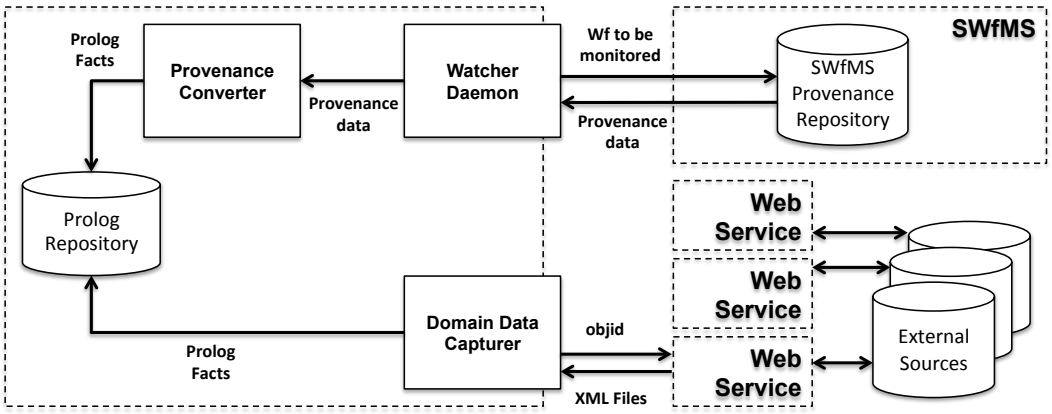

Fig. 2. Architecture showing how to convert, import and export provenance and domain data. Provenance extracted from SWfMS provenance databases as well as external sources are converted into Prolog facts and stores in a central Prolog Repository.

provenance. This database may contain prospective and retrospective provenance data as well as implicit provenance captured on file system level [Oliveira et al. 2014].

The main challenge of our architecture resides in providing an integration scheme that is generic enough to work with different SWfMS. To do so, our architecture provides the Watcher Daemon, which is able to connect to a given provenance DBMS and to provide data to the Provenance Converter that generates Prolog Facts. The *daemon* works as a cartridge that connects to a given DBMS. For this article, we implemented a cartridge for the PostgreSQL, since the SWfMS we use in our use case (SciCumulus) use this DBMS.

For each new workflow execution, new tuples related to prospective and retrospective provenance of this execution are inserted in the several tables of the database, and the *daemon* is invoked. The only parameter value that is important for the *daemon* is the key attribute that identifies a workflow execution in the database (*e.g. ewkfid* in *eWorkflow* table in the case of the SciCumulus' provenance database). Once the *daemon* identifies new provenance data in the database, it calls the **Provenance Converter** to convert all tuples of interest (from several different tables) into Prolog facts.

The **Provenance Converter** receives tuples from the **Watcher Daemon** and converts them into Prolog facts. Then, the facts are stored in the Prolog repository. For each relational table tuple, a Prolog fact of the form $TableName(Att_1Value, Att_2Value, ..., Att_nValue)$ is created. The Provenance Converter also defines a predicate (*i.e.* atom) for each of the tables of the relational schema (*i.e.* the name of the predicate is the same as the table name). To illustrate, Table I shows the Atom structure and examples of some facts extracted from the relational database which schema is presented in Fig. 3.

External Sources are repositories that provide domain-specific data (*e.g.*, from bioinformatics) and can correspond to several domain-specific datasets available on the Web such as EBI, NCBI, and GenBank. They can be used to enrich provenance data information about organism lineage, sequence length, molecule type, *etc.* However, external sources are usually huge databases that would take a long time to download, and require lots of storage space. Thus, importing entire external datasets is not an option. This way, we import just a subset of external datasets by using a filter based on object ids (*objid*) of interest. As an example, we may import functions from GenBank related to a specific gene or organisms' taxonomic division. This is done by the **Domain Data Capturer**.

The **Domain Data Capturer** component receives an *objid* of interest (*e.g.*, biological sequence identification) and performs Web Service calls to external data sources to retrieve domain data about that object. The Web Service returns the desired domain data represented in XML that is converted into Prolog facts by the **Domain Data Capturer** component using the approach of Lima et al.

Table I. Atom Structure and associated facts extracted from SciCumulus provenance database.

Atom structure	**Facts**
cworkflow(wkfid, tag).	cworkflow(13, 'SciPhy'). cworkflow(14, 'SciEvol').
eworkflow(ewkfid, tag, tagexec).	eworkflow(36, 'SciPhy', 'SciPhy_1'). eworkflow(37, 'SciPhy', 'SciPhy_2'). eworkflow(38, 'SciEvol', 'SciEvol_1'). eworkflow(39, 'SciEvol', 'SciEvol_2').
cactivity(actid, wkfid, tag, status).	cactivity(525, 13, 'act1'). cactivity(526, 13, 'act2'). cactivity(527, 14, 'act1'). cactivity(528, 14, 'act2').
eactivity(actid, ewkfid, cactid, tag, status, starttime, endtime).	eactivity(132, 36, 525, 'act1', 'running', 2016-02-22 20:29:31, 2016-02-22 21:18:27). eactivity(133, 37, 526, 'act2', 'running', 2016-02-22 21:18:27, 2016-02-22 22:23:13). eactivity(134, 38, 527, 'act1', 'finished', 2016-02-23 22:23:13, 2016-02-23 22:59:09). eactivity(135, 39, 528, 'act2', 'finished', 2016-02-23 22:59:09, 2016-02-24 00:37:16).
eactivation(taskid, actid, status, duration).	eactivation(89, 132, 'running', 4). eactivation(90, 132, 'running', 3). eactivation(91, 132, 'running', 1). eactivation(92, 132, 'running', 2). eactivation(93, 133, 'finished', 30). eactivation(94, 133, 'finished', 21). eactivation(95, 133, 'finished', 2). eactivation(96, 133, 'finished', 8).
imod(ik, ewkfid, eactid, taskid, path, file, seqid).	imod(1, 36, 132, 89, '/root/SciPhy/fasta1', 'fasta1', 'A00145'). imod(2, 36, 132, 90, '/root/SciPhy/fasta2', 'fasta2', 'A00144'). imod(3, 36, 132, 91, '/root/SciPhy/fasta3', 'fasta3', 'A00143'). imod(4, 36, 133, 96, '/root/SciPhy/fasta3', 'fasta3', 'A00142').
omod(ok, ik, ewkfid, eactid, taskid, path, file).	omod(1, 1, 36, 132, 89, '/root/SciPhy/fasta2', 'fasta2'). omod(2, 2, 36, 132, 90, '/root/SciPhy/fasta3', 'fasta3'). omod(3, 3, 36, 132, 91, '/root/SciPhy/fasta4', 'fasta4').
cfield(fname, relid, ftype).	cfield('name', 1, 'String'). cfield('description', 2, 'String'). cfield('size', 3, 'Integer'). cfield('index', 4, 'Integer').
crelation(relid, actid, rtype, rname).	crelation(1, 525,'input', 'idataselection'). crelation(2, 526,'input', 'imafft'). crelation(3, 527,'input', 'ireadseq'). crelation(4, 528,'input', 'iraxml').
tree(trid, bootstrap).	tree(1,86). tree(2,72). tree(3,83).
treeSeq(trid, seqid).	treeSeq(1,'A00145'). treeSeq(2,'A00144'). treeSeq(3,'A00143'). treeSeq(4,'A00142').

[2012]. Then, it stores the Prolog facts in the **Prolog Repository**. Thus, the **Prolog Repository** integrates provenance data with domain data that can be easily queried. Since all data is in the repository, a specialist user could implement and use Prolog rules to submit their queries. The only requirement is that the external source must be accessible through Web services (or other kind of query that provides XML as response). We call these services to retrieve a subset of interest of these datasets. Web services of this type are available for several important biological datasets, and also for datasets of other domains [Stein 2003; Amirian and Alesheikh 2008]. Note that domain data can be imported both during and after the workflow execution, and thus our approach allows queries to be performed at both of these moments.

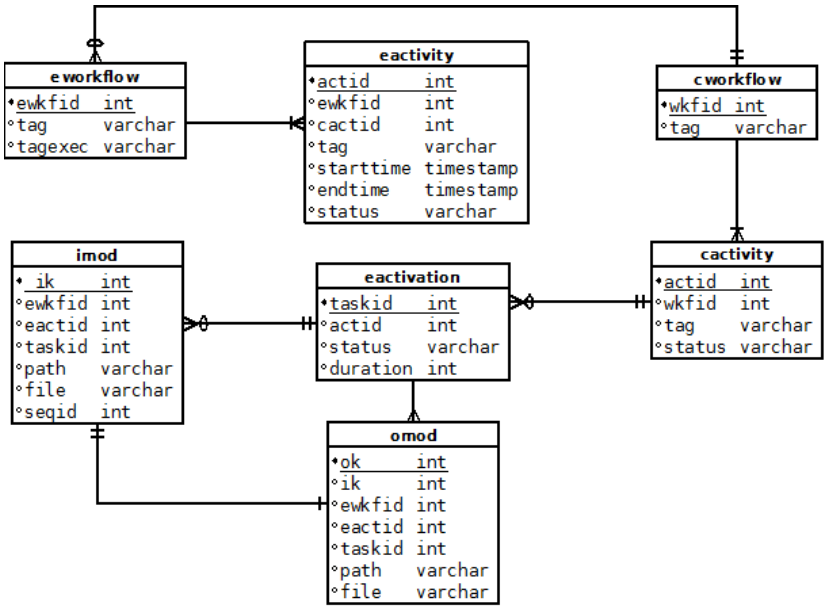

Fig. 3. Part of the SciCumulus provenance database schema.

We implemented our architecture and coupled it with the SciCumulus SWfMS. However, our architecture is designed to be generic and able to be applied to most of the SWfMS. Note that our Prolog converter is able to read the database structure and convert it into Prolog. However, it is worth noticing that for most existing SWfMS, the proposed approach will only work after the workflow execution, since Provenance data is provided at the end of the execution (e.g. Swift/T, VisTrails). If the SWfMS is able to provide Provenance data during the workflow execution (e.g. SciCumulus and Pegasus), then our approach is able to generate Prolog rules at runtime. In any case, this does not compromise the usefulness of our approach.

4.2 Provenance Knowledge Base

Our provenance knowledge base is comprised of facts and rules extracted from the provenance database and external domain data. It is represented in Prolog and can be queried at any moment. Our Prolog representation uses the same structure provided by the SWfMS provenance database. This means that it may not be compliant with a provenance model. It all depends on how the SWfMS chose to represent its data. Table I illustrates an excerpt of facts and rules extracted from the SciCumulus provenance database, while Table II illustrates data extracted from external databases.

Although we believe compliance with a provenance model is very important, in this work we opted for abstracting this requirement, since, in practice, this would make our approach to work only with a limited number of SWfMS. If needed, one can map the provenance database into one of the available provenance models, and then export the resulting mapping into Prolog. We give some first steps in this direction in a preliminary work [Oliveira et al. 2016].

We have implemented our architecture using Java. The *daemon* and the *domain data capturer* are generic. For this implementation, the *daemon* works with the PostgreSQL DBMS, and the Domain Data Capturer works with the EBI Web Services. The Prolog query interface was implemented using SWI.

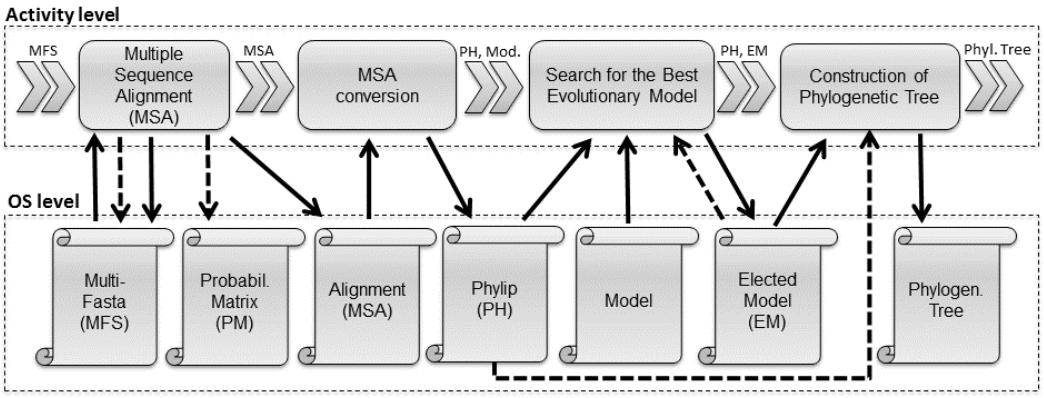

Fig. 4. An abstract representation of the SciPhy workflow with four activities: Sequence Alignment, Conversion of Format, Search of Evolutionary Model and Construction of the Phylogenetic Tree.

4.3 Usage Example

We have chosen the SciPhy workflow [Ocaña et al. 2011] to illustrate how our approach could be used in practice. SciPhy is a scientific workflow that aims at generating phylogenetic trees from DNA, RNA, or amino acid sequences. Phylogenetic trees determine the inferred evolutionary relationships among various biological species. An abstract representation of SciPhy is presented in Fig. 4. SciPhy is composed of four activities. They are responsible for executing the phylogenetic analysis (or gene phylogeny) and are named as: (a) Multiple Sequence Alignment (MSA), (b) MSA conversion, (c) Search for the Best Evolutionary Model, and (d) Construction of the Phylogenetic Tree. Each activity is associated with a specific program. Multiple sequence alignment activity may be implemented by MAFFT [Katoh and Toh 2010], Kalign [Lassmann et al. 2009], ClustalW [Larkin et al. 2007] or ProbCons [Do et al. 2005]. Each MSA program receives a multi-fasta file as input (MFS in Fig. 4), then produces a MSA as output (MSA in Fig. 4). Multi-fasta is a text-based format for representing either nucleotide sequences or peptide sequences, in which nucleotides or amino acids are represented using single-letter codes. Besides producing the MSA file, some programs such as ClustalW produce auxiliary files such as a probabilistic matrix that can be used for reconstructing the sequences (PM in Fig. 4). PM is produced by ClustalW, but it is not informed by the program, *i.e.* scientists only discover that this auxiliary file exists if they search on the file directory (which justifies capturing implicit provenance data). Following, the MSA conversion is implemented by ReadSeq [Gilbert 2002]. ReadSeq receives the MSA in different formats (one for each MSA program) and then converts it to PHYLIP format [Felsenstein 2006] (PH in Fig. 4). The election of the evolutionary model is implemented by ModelGenerator [Keane et al. 2006]. There are several different evolutionary models available, and ModelGenerator chooses the best one (EM in Fig. 4) to use in the construction of the phylogenetic tree by RAxML [Stamatakis 2006] (Phylogenetic Tree in Fig. 4). In SciPhy, we consider pre-chosen evolutionary models (BLOSUM62, CPREV, JTT, WAG, or RtREV) and we obtain several trees for each one of the MSA programs used.

As presented in Fig. 4, the files MFS, MSA, PM, PH, Model, EM, and Phylogenetic Tree are created, accessed or modified by one or more activities during the workflow execution. Dotted arrows among activities and files denote data flows not explicitly declared in the workflow specification (implicit retrospective provenance) while solid arrows denote data flows declared in the workflow specification (explicit retrospective provenance).

We used the SciCumulus SWfMS [Oliveira et al. 2010] to run SciPhy in our case study, and so the provenance repository we use is the one generated by SciCumulus. SciCumulus stores the provenance data and (domain-specific) data extracted from produced files in a PostgreSQL relational database. A key point here is that the domain data that SciCumulus is able to store is extracted from the files

Table II. Atom structure and associated facts extracted from part of the EBI biological database

Atom Structure	**Facts**
accession(entryid, accid, tag).	accession(149, 150, 'A00145').
taxonomicdivision(entryid, txid, tag).	taxonomicdivision(149, 158, 'MAM').
sequencelength(entryid, idlength, length).	sequencelength(149, 162, '678').
scientificname(taxonid, scinameid, name).	scientificname(204, 205, 'Bos taurus'). scientificname(212, 213, 'Eukaryota'). scientificname(215, 216, 'Metazoa'). scientificname(218, 219, 'Chordata'). scientificname(221, 222, 'Craniata').

that are used as input and output to the workflow. No external domain data is supported. Fig. 3 shows part of the schema of the SciCumulus provenance database. The complete schema is available at `http://www.ic.uff.br/~vanessa/papers/relational-database-schema-SCC.png`.

In this schema we can identify a series of tables that represent the workflow structure. Table *cworkflow* stores metadata about the workflow, and *cactivity* stores metadata about activities of a workflow. It also has tables that represent the workflow execution. Table *eworkflow* stores metadata about the execution of a workflow, *eactivity* stores metadata about the execution of an activity and *eactivation* stores metadata about the execution of tasks of an activity, each one consuming a portion of the data. Finally, the schema is able to store domain-specific data that is extracted from input and output data files. In this case, the tables will have names and structure that are specific for each workflow – this is automatically handled by SciCumulus. As an example, table *imod* stores the name of the input file used by an activation and the *sequence identification (seqid)* that is within this file, and table *omod* holds the name of the output files generated by an activation.

Before running the workflow, the scientist starts the *daemon* and tells it to monitor the experiment execution by passing the SciPhy *tagexec*. Then, she/he runs the workflow normally using SciCumulus. During the execution, the *daemon* watches the database, collecting tuples and sending them to the Provenance Converter. These tuples are then stored in the Prolog Repository as Prolog Facts. Every time a tuple is sent to the Provenance Converter, it first checks if there is an atom corresponding to the structure of that tuple. If not, it creates it. Table I shows a fragment of the Prolog facts that were generated by Provenance Converter for this case study. The entire set of facts can be downloaded at `http://www.ic.uff.br/~vanessa/papers/Prolog.rar`.

After the first facts are in the Prolog Repository, the scientist can start analyzing the experiment by posing Prolog queries. Since there are several experiments that take a long time to complete (days or even weeks), waiting for it to finish may represent wasting precious computational resources, since errors would be detected too late. Our approach allows for online analysis, and thus completely avoids this problem.

At any time (during or after the experiment execution), the scientist may decide to load external domain data into the provenance repository. In the case of the SciPhy workflow, the scientist may retrieve metadata related to the DNA, RNA or amino acid sequences to support provenance analysis. The SciCumulus provenance database has a table named *imod* (shown in Fig. 3) that holds the *seqid* attribute. This attribute is extracted from input data files of the SciPhy workflow. However, the *seqid* itself does not bring much information for scientists. Thus, for each tuple of the *imod* table related to the chosen workflow execution (*tagexec*) of SciPhy, the Domain Data Capturer calls the EBI RESTFul Web Service passing the *seqid* (*objid* in our architecture) as a parameter. External public biological databases hold important information such as organism lineage, sequence length, molecule type, taxonomic division, *etc.* Such information is converted into Prolog facts by the Domain Data Capturer and stored in the Prolog Repository. Table II shows a list of some atoms and facts extracted from the XML file generated by EBI Web services. Once external data is imported into the provenance

Table III. Prolog Rules that can be used by scientists to query the provenance database. These rules show the potential of our proposed approach in querying prospective, retrospective and domain data, during and after the workflow execution.

Rule	PC	Query in PC	Rule
1	3	1,5	influency(Influencer, Influenced) :- imod(Ik,_,_,_,_,Influencer,_), omod(_,Ik,_,_,_,_,Influenced). influencyChain(Influencer, Influenced) :- influency(Influencer, Influenced). influencyChain(Influencer, Influenced) :- influency(Influencer, Any),influencyChain(Any, Influenced).
2	1,2	1,2,3(1); 1,2,3(2)	consistentTrees(WkfEx,TreeId) :- tree(TreeId,Bootstrap), Bootstrap > 80, treeSeq(TreeId, SeqId), eworkflow(WkfEx,_,_), imod(_,WkfEx,_,_,_,_,SeqId).
3	1,2	1,2,3(1); 1,2,3(2)	taxDivDur(Organism,TaxDiv,Duration) :- feature(E,F), taxonomicdivision(E,_,TaxDiv), taxon(F,T1), scientificname(T1,_,Organism), accession(E,_,SeqId), imod(_,_,_,Taskid,_,_,SeqId), eactivation(Taskid,_,_,Duration).
4	1,2	1,2,3(1); 1,2,3(2)	wkfActExec(TaxDiv,TagExec,Tag,Status) :- taxonomicdivision(E,_,TaxDiv), accession(E, _, SeqId), imod(_,_,Eactid,_,_,_,SeqId), eactivity(Eactid,Wkfid,_,Tag,Status), eworkflow(Wkfid,_,TagExec).
5	–	–	lineageOrg(Organism, SciName) :- taxon(_,T1), scientificname(T1,_, Organism), lineage(T1,L), taxon(L,T2), scientificname(T2,_,SciName).
6	–	–	seqLength(SeqId,Size) :- accession(E,_,SeqId), sequencelength(E,_,Size).
7	3	1,5	treeTaxa(Taxon, TreeId, Param) :- feature(En,FetId), taxon(FetId,TaxonId), scientificname(TaxonId,_,Taxon), accession(En, _,SeqID), imod(_,WkfEx,ActEx,_,_,_,SeqId), tree(TreeId,_), treeSeq(TreeId, SeqId), cactivity(Actid,_,_), eactivity(ActEx, WkfEx, Actid,_,_), cfield(Param, Relid,_), crelation(Relid, Actid, 'input',_).

database as Prolog facts, scientists can start submitting Prolog queries that combine provenance and external data.

5. CASE STUDY: RELATING PROVENANCE AND EXTERNAL DOMAIN DATA

Our case study shows how the scientist can take advantage of the integration of provenance and external domain data to execute queries that otherwise would not be able to be performed. The workflow we use is SciPhy (described in Section 4.3). Queries presented in this section were inspired by the queries of the First, Second and Third Provenance Challenges[1]. Table III shows seven Prolog rules that can be used to infer knowledge about the phylogenetic analysis experiment. The table also shows the Provenance Challenge in which the query was inspired (column PC), and which query from which Provenance Challenge inspired that specific query (column Query in PC). A Computer Science specialist user can add other Prolog rules as needed. These rules aim at allowing scientists to relate different types and granularity levels of provenance data and combine them with domain data obtained from EBI and other external sources. This way, they offer an abstraction of the structure of the Prolog facts, thus allowing scientists to use them to query provenance without worrying about the data schema and the complex relationships in the context of each rule. To use them, all the scientist need to do is to inform at least one parameter value. If no parameter value is informed, then all possible combinations are returned by the query.

To answer the query "Which files influenced the generation of the *fasta4* output file?", the scientist can use Rule 1 shown in Table III. This rule was inspired by noWorkflow [Murta et al. 2014]. It recursively retrieves all input files that influenced the generation of a particular output file (*i.e.*, transitive closure). To do so, it explores retrospective provenance. If some file references are not explicitly declared in the workflow activity (as parameters, for example), this query is also able to

[1]Provenance Challenge is the official event of the provenance community to compare and understand existing provenance-aware approaches (`http://twiki.ipaw.info/bin/view/Challenge/WebHome`).

```
?- influencyChain(File, 'fasta4').

File = fasta3;
File = fasta1;
File = fasta2.
```

Fig. 5. Result of the query InfluencyChain for the *fasta4* file

```
?- consistentTrees(WkfEx, TreeId).

WkfEx = 36, TreeId = 1;
WkfEx = 36, TreeId = 3.
```

Fig. 6. Result of the query consistentTrees

```
?- taxDivDur('Bos taurus', TaxDiv, Duration).

TaxDiv = 'MAM', Duration = 4.
```

Fig. 7. Result of the query taxDivDur.

retrieve the results. This situation is common in SciPhy since MSA programs (such as ClustalW) produce the alignment but also produce other files that are important for bioinformaticians such as the probabilistic matrix (PM in Fig. 4) that is important for sequence reconstruction and for validating statistical data in phylogenetic trees. Thus, in the case of SciPhy, the query also explores implicit provenance. To pose this query, the scientist fills in its second parameter as *fasta4*. The query *influencyChain(File, 'fasta4')*, returns the names of all files that contributed to the generation of the *fasta4* file (Fig. 5). This type of query is extremely important to scientists to discover which files influenced the results of a specific experiment. With this information, they can check if each activity processed the correct files. They may also find applications/services errors even when the scientist is unaware of the source code (that is, even when the activity is a black-box). Thus, scientists are able to verify if mistakes made over intermediate results influenced the final result and reproduce the workflow with greater knowledge about its behavior.

Rule 2 implements a query proposed in the Introduction. It is designed to retrieve the most consistent trees ($bootstrap > 80$) for each workflow execution ($WkfEx$). It explores explicit retrospective provenance. The result is presented in Fig. 6.

These two previous queries did not use external data. We now present some queries that can only be answered when external data is integrated into the provenance database. Suppose the scientist wants to know how long the execution of activities that use sequences related to the *Bos taurus* organism lasted, along with its taxonomy division. To do this, she/he can use Rule 3. This rule allows for querying the taxonomic division and the time consumed by each task that processes data related to an organism. It is an example of integration between information gathered from external data sources (the taxonomic division) and explicit retrospective provenance data (the duration of activities). It returns specific domain data that may help non-expert or beginners for better understanding provenance data related to a particular organism. To answer this query, the scientist can perform the query *taxDivDur('Bos taurus', TaxDiv, Duration)* and know about the taxonomic divisions (*e.g.*, MAM - Mammalia or PRI - Primate), which comes from the EBI databases, as well as the time spent in the task executions (in seconds). Fig. 7 shows the query result.

To get the status of a workflow experiment that is experimenting with sequence IDs related to a given taxonomic division, the scientist can use Rule 4. This query uses external domain data and also explicit prospective and retrospective provenance. Since it retrieves the workflow ID (which is referred to as tag in SciCumulus), the scientist can further explore it by using other queries later. The results of a query for the taxonomic division *MAM*, which refers to Mammalia, are shown in Fig. 8. The facts *taxon*, *lineage*, *taxonomicdivision*, and *scientificname* used in the rule body (Table III) come from the EBI database. Example facts are listed in Table II.

In order to retrieve the organism's lineage, scientists can benefit from Rule 5. For that, they must inform the organism name as the input parameter. Passing the organism name *Bos taurus* as a parameter, for instance, the query returns its respective lineage: *Eukaryota, Metazoa, Chordata, Craniata, Vertebrata, Euteleostomi, Mammalia, Eutheria, Laurasiatheria, Cetartiodactyla, Ruminantia, Pecora,*

```
?- wkfActExec('MAM',TagExec,Tag,Status).

TagExec = 'SciPhy_3', Tag = act1, Status = running;
TagExec = 'SciPhy_4', Tag = act2, Status = running.
```

Fig. 8. Result of the query wkfActExec.

```
?- lineageOrg('Bos taurus',SciName).

SciName = 'Eukaryota'; SciName = 'Metazoa'; SciName = 'Chordata'; SciName = 'Craniata'; SciName = 'Vertebrata';
SciName = 'Euteleostomi'; SciName = 'Mammalia'; SciName = 'Eutheria'; SciName = 'Laurasiatheria'; SciName =
'Cetartiodactyla'; SciName = 'Ruminantia'; SciName = 'Pecora'; SciName = 'Bovidae'; SciName = 'Bovinae'; SciName
= 'Bos'.
```

Fig. 9. Result of the query lineageOrg.

```
?- seqLength('A00145', Size).

Size = '678'.
```

Fig. 10. Result of the query seqLength.

```
?- treeTaxa('Bos taurus', TreeId, Param).

TreeId = 1, Param = name;
TreeId = 1, Param = description.
```

Fig. 11. Result of the query treeTaxa.

Bovidae, Bovinae, and *Bos*. This information may help scientists to identify which organisms are classified into the same lineage (or in the same phylogenetic tree) and enables performing queries - the same or other - over the requested organisms. Fig. 9 shows the lineage results to the *'Bos taurus'* organism. The facts *accession* and *sequencelength* used here come from the EBI database and can be seen in Table II. This is an example of query that explores only external data. No crossing with provenance data is made.

To obtain the length of a gene sequence the scientist can use Rule 6 by passing the sequence identifier (*SeqId*) as a parameter as shown in Fig. 10. For example, running a query using this rule with the *SeqId* A00145, we obtain the value 678 for this specific gene sequence. Similarly to Rule 5, this rule queries only external domain data.

Finally, Rule 7 implements the second query proposed in the Introduction: Which trees contain the taxon *Bos taurus* and which parameters were used to generate those trees? It is designed to retrieve the trees that have a specific taxon and the parameters used to derive them. For this, it crosses explicit retrospective provenance and external domain data. The result produced from that query is presented in Fig. 11.

This case study aimed at evaluating the suitability of the proposed approach. In fact, new analyses are still needed, such as scalability ones with large volumes of imported data. These new experiments are already planned as future work.

6. FINAL REMARKS

Analyzing provenance data is still an open, yet fundamental problem that deserves special attention from the scientific community. The challenge is even more complex when scientists need to correlate provenance and domain-specific data. To deal with this problem, we present an approach that allows for scientists to analyze different types of provenance data together with external domain data. Although there are several efforts to enrich provenance with domain data [Fileto et al. 2003; Gonçalves et al. 2012; Oliveira et al. 2015], our approach contributes by allowing external data to be queried together with provenance database, even when the need for external data was not anticipated during the workflow design and execution. Our case study showed the feasibility of the approach in analyzing provenance through Prolog queries. This approach is generic and can be applied on provenance repositories generated by different provenance capturing systems.

In the case study presented in this article, we developed a Web service client to call the EBI Web service. The Domain Data Capturer calls the Web Service and converts the obtained XML into Prolog facts using the approach developed by Lima et al. [2012]. This Web service client can be extended to query other external sources such as NCBI and UniProtKB/Swiss-Prot. We also developed Prolog rules to integrate facts extracted from the SciCumulus provenance database and the EBI Web service. Scientists can benefit from our proposed approach for relevant bioinformatics processes such as sequence annotation and experiment validation. This way, they can associate a specific *SeqId* to a specific activity execution in their experiments. They can also link this information to several other external databases. This allows them to deepen their knowledge about a specific domain such as phylogenetic analysis, performing recursive and non-recursive queries without worrying about the specific aspects of the Prolog language. In practice, depending on the domain, new rules would be needed by the scientists. Computer Science experts can support different requirements by creating other Prolog rules and adding them to the Prolog Repository. This can be made either during or after the workflow execution.

Our current implementation relies on the domain data extractor provided from SciCumulus. This extractor is able to look into the input and output files of the experiment and extract relevant domain information that is then stored in its provenance database. This is the case of the *imod* and *omod* tables we used in our case study. However, few SWfMS are capable of doing this – most of them only store the files and do not look into their content. Thus, when plugging our approach into other SWfMS, one would need to develop domain extractors [Gonçalves et al. 2012]. This is needed so that we can connect provenance generated by the workflow with external domain data by calling Web services with relevant parameters (*objid* in our architecture).

As future work, we plan to add experiments from other domains, such as astronomy (Montage Workflow). We also plan to make querying easier by providing an interface where the user would be able to select attributes, and the system would automatically generate the corresponding Prolog query. This would eliminate the need of a Computer Scientist to write the rules in our current approach. An initial idea is discussed by Martins et al. [2013].

We also plan to extend our approach by allowing scientists to connect more than one provenance database into the system. This way, collaborative research teams that work on similar experiments can share their findings and benefit from results of similar workflow executions. A first step in this direction has been given by providing an approach to query heterogeneous provenance databases [Oliveira et al. 2016].

REFERENCES

Abu-Asab, M. S., Chaouchi, M., Alesci, S., Galli, S., Laassri, M., Cheema, A. K., Atouf, F., VanMeter, J., and Amri, H. Biomarkers in the Age of Omics: Time for a Systems Biology Approach. *OMICS: A Journal of Integrative Biology* 15 (3): 105–112, 2011.

Alper, P., Belhajjame, K., Goble, C. A., and Karagoz, P. Enhancing and abstracting scientific workflow provenance for data publishing. In *International Workshop on Managing and Querying Provenance Data at Scale (BigProv)*. ACM, New York, NY, USA, pp. 313–318, 2013.

Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., and Mock, S. Kepler: an extensible system for design and execution of scientific workflows. In *Scientific and Statistical Database Management Conference (SSBDM)*. Santorini Island, Greece, pp. 423–424, 2004.

Amirian, P. and Alesheikh, A. A. Publishing Geospatial Data through Geospatial Web Service and XML Database System. *American Journal of Applied Sciences* 5 (10): 1358–1368, Jan., 2008.

Buttler, D., Coleman, M., Critchlow, T., Fileto, R., Han, W., Pu, C., Rocco, D., and Xiong, L. Querying Multiple Bioinformatics Information Sources: can Semantic Web research help? *SIGMOD Record* 31 (4): 59–64, 2002.

Callahan, S. P., Freire, J., Santos, E., Scheidegger, C. E., Silva, C. T., and Vo, H. T. Vistrails: Visualization meets data management. In *International Conference on Management of Data (SIGMOD)*. ACM, New York, NY, USA, pp. 745–747, 2006.

Carvalho, M. G. P. d., Campos, L. M., Braganholo, V., Campos, M. L. M., and Campos, M. L. A. Extracting New Relations to Improve Ontology Reuse. *Journal of Information and Data Management (JIDM)* 2 (3): 541, 2011.

Ceri, S., Gottlob, G., and Tanca, L. What you Always Wanted to Know about Datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 1 (1): 146–166, 1989.

Chong, S. Logic programming. Computing Science Technical Report lec19, Harvard School of Engineering and Applied Sciences, Cambridge, MA, 2016.

Costa, F., De Oliveira, D., Ocala, K., Ogasawara, E., Dias, J., and Mattoso, M. Handling failures in parallel scientific workflows using clouds. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*. Salt Lake City, UT, USA, pp. 129–139, 2012.

Deelman, E., Vahi, K., Rynge, M., Juve, G., Mayani, R., and da Silva, R. Pegasus in the cloud: Science automation through workflow technologies. *IEEE Internet Computing* 20 (1): 70–76, Jan, 2016.

Dey, S., Köhler, S., Bowers, S., and Ludäscher, B. Datalog as a lingua franca for provenance querying and reasoning. In *USENIX Conference on Theory and Practice of Provenance (TaPP)*. USENIX Association, Berkeley, CA, USA, pp. 13–13, 2012.

Do, C. B., Mahabhashyam, M. S. P., Brudno, M., and Batzoglou, S. ProbCons: Probabilistic consistency-based multiple sequence alignment. *Genome Research* 15 (2): 330–340, 2005.

Doolittle, W. F. Phylogenetic classification and the universal tree. *Science* 284 (5423): 2124–2128, Oct., 1999.

El-Jaick, D., Lima, A. A. B., and Mattoso, M. SGProv: summarization mechanism for multiple provenance graphs. *Journal of Information and Data Management (JIDM)* 5 (1): 16–27, 2014.

Ellqvist, T., Koop, D., Freire, J., and Silva, C. Using mediation to achieve provenance interoperability. In *2009 World Conference on Services - I*. Los Angeles, CA, pp. 291–298, 2009.

Felsenstein, J. Phylogeny inference package. Available at `http://fumblog.um.ac.ir/gallery/309/PHYLIP.pdf`, 2006.

Fileto, R., Medeiros, C. B., Liu, L., Pu, C., and Assad, E. D. Using domain ontologies to help track data provenance. In *Brazilian Symposium on Databases (SBBD)*. Vol. 18. Manaus, Brazil, pp. 84–98, 2003.

Freire, J., Koop, D., Santos, E., and Silva, C. Provenance for Computational Tasks: a survey. *Computing in Science & Engineering* 10 (3): 11–21, May, 2008.

Futuyma, D. J. *Evolution*. Sinauer Associates, Inc, 2013.

Gilbert, D. Sequence File Format Conversion with Command-Line Readseq. In *Current Protocols in Bioinformatics*. John Wiley & Sons, Inc., Hoboken, NJ, USA, pp. A. 1E. 1 – A. 1E. 4, 2002.

Gonçalves, G. C., de Oliveira, D., Ocaña, K., Ogasawara, E., and Mattoso, M. Using Domain-Specific Data to Enhance Scientific Workflow Steering Queries. In *International Provenance and Annotation Workshop (IPAW)*. Springer, Santa Barbara, CA, pp. 152–167, 2012.

Halevy, A., Rajaraman, A., and Ordille, J. Data Integration: the teenage years. In *International Conference on Very Large Data Bases (VLDB)*. ACM, Seoul, Korea, pp. 9–16, 2006.

Katoh, K. and Standley, D. M. MAFFT Multiple Sequence Alignment Software Version 7: Improvements in Performance and Usability. *Molecular Biology and Evolution* vol. 30, pp. 772–780, 2013.

Katoh, K. and Toh, H. Parallelization of the MAFFT Multiple Sequence Alignment Program. *Bioinformatics* 26 (15): 1899–1900, 2010.

Keane, T. M., Creevey, C. J., Pentony, M. M., Naughton, T. J., and Mclnerney, J. O. Assessment of Methods for Amino Acid Matrix Selection and their use on Empirical Data shows that ad hoc Assumptions for Choice of Matrix are not Justified. *BMC Evolutionary Biology* 6 (29): 1471–2148, 2006.

Larkin, M. A., Blackshields, G., Brown, N. P., Chenna, R., McGettigan, P. A., McWilliam, H., Valentin, F., Wallace, I. M., Wilm, A., Lopez, R., Thompson, J. D., Gibson, T. J., and Higgins, D. G. Clustal W and Clustal X version 2.0. *Bioinformatics* 23 (21): 2947–2948, 2007.

Lassmann, T., Frings, O., and Sonnhammer, E. L. L. Kalign2: high-performance multiple alignment of protein and nucleotide sequences allowing external features. *Nucleic Acids Research* 37 (3): 858–865, 2009.

Lima, D., Delgado, C., Murta, L., and Braganholo, V. Towards Querying Implicit Knowledge in XML Documents. *Journal of Information and Data Management (JIDM)* 3 (1): 51–60, Oct., 2012.

Marinho, A., Werner, C., Mattoso, M. L. Q., Braganholo, V., and Murta, L. G. P. Challenges in Managing Implicit and Abstract Provenance Data: experiences with ProvManager. In *Workshop on the Theory and Practice of Provenance (TaPP)*. Usenix, Herakli Crete, Greece, pp. 1–6, 2011.

Martins, G., Larcher Junior, C., Oliveira, A., Murta, L., and Braganholo, V. XChange: Compreensão de Mudanças em Documentos XML. In *Sessão de Demos do Simpósio Brasileiro de Bancos de Dados (SBBD)*. Recife, Brasil, 2013.

Mattoso, M., Dias, J., na, K. A. O., Ogasawara, E., Costa, F., Horta, F., Silva, V., and de Oliveira, D. Dynamic steering of {HPC} scientific workflows: A survey. *Future Generation Computer Systems* vol. 46, pp. 100 – 113, 2015.

Mattoso, M., Werner, C., Travassos, G. H., Braganholo, V., Ogsawara, E., de Oliveira, D., da Cruz, S. M. S., Martinho, W., and Murta, L. Towards Supporting the Life Cycle of Large Scale Scientific Experiments. *International Journal of Business Process Integration and Management* 5 (1): 79–92, Oct., 2010.

Moreau, L., Freire, J., Futrelle, J., McGrath, R. E., Myers, J., and Paulson, P. The Open Provenance Model: An Overview. In *International Provenance and Annotation Workshop (IPAW)*. pp. 323–326, 2008.

Moreau, L., Missier, P., Belhajjame, K., B'Far, R., Cheney, J., Coppens, S., Cresswell, S., Gil, Y., Groth, P., Klyne, G., Lebo, T., McCusker, J., Miles, S., Myers, J., Sahoo, S., and Tilmes, C. PROV-DM: The PROV data model. `http://www.w3.org/TR/2013/REC-prov-dm-20130430/`, 2013.

Murta, L., Braganholo, V., Chirigati, F., Koop, D., and Freire, J. noworkflow: Capturing and analyzing provenance of scripts. In *International Provenance and Annotation Workshop (IPAW)*. Springer, Cologne, Germany, pp. 71–83, 2014.

Ocaña, K., de Oliveira, D., Ogasawara, E., Dávila, A., Lima, A., and Mattoso, M. SciPhy: a cloud-based workflow for phylogenetic analysis of drug targets in protozoan genomes. In *Advances in Bioinformatics and Computational Biology*, O. N. de Souza, G. P. Telles, and M. Palakal (Eds.). Lecture Notes in Computer Science, vol. 6832. Springer, Brasília, Brazil, pp. 66–70, 2011.

Ogasawara, E., Dias, J., Silva, V., Chirigati, F., de Oliveira, D., Porto, F., Valduriez, P., and Mattoso, M. Chiron: a parallel engine for algebraic scientific workflows. *Concurrency and Computation: Practice and Experience* 25 (16): 2327–2341, 2013.

Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M. R., Wipat, A., and Li, P. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics* 20 (17): 3045–3054, 2004.

Oliveira, D., Ocaña, K. A. C. S., Baião, F., and Mattoso, M. A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds. *Journal of Grid Computing* 10 (3): 521–552, 2012.

Oliveira, D., Silva, V., and Mattoso, M. How much domain data should be in provenance databases? In *Workshop on the Theory and Practice of Provenance (TaPP)*. USENIX Association, Edinburgh, Scotland, 2015.

Oliveira, D. C. M. d., Ogasawara, E., Baião, F., and Mattoso, M. Scicumulus: A lightweight cloud middleware to explore many task computing paradigm in scientific workflows. In *IEEE International Conference on Cloud Computing (CLOUD)*. Miami, USA, pp. 378–385, 2010.

Oliveira, W., Missier, P., Ocaña, K., de Oliveira, D., and Braganholo, V. Analyzing Provenance across Heterogeneous Provenance Graphs. In *International Provenance and Annotation Workshop (IPAW)*. Washington D.C., USA, pp. 57–70, 2016.

Oliveira, W., Neves, V., Ocaña, K., Murta, L., Oliveira, D. d., and Braganholo, V. Captura e Consulta a Dados de Proveniência Retrospectiva Implícita Intra-Atividade. In *Simpósio Brasileiro de Banco de Dados (SBBD)*. Curitiba, PR, Brazil, pp. 35–44, 2014.

Sahoo, S. S., Nguyen, V., Bodenreider, O., Parikh, P., Minning, T., and Sheth, A. P. A Unified Framework for Managing Provenance Information in Translational Research. *BMC Bioinformatics* 12 (1): 461, 2011.

Sahoo, S. S., Weatherly, B., Mutharaju, R., Anantharam, P., Sheth, A. P., and Tarleton, R. L. Ontology-driven Provenance Management in eScience: An application in parasite research. In *International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE)*. Portugal, 2009.

Searls, D. B. Pharmacophylogenomics: genes, evolution and drug targets. *Nature Reviews Drug Discovery* 2 (8): 613–623, Aug., 2003.

Seltzer, M. Provenance integration requires reconciliation. In *International Workshop on Theory and Practice of Provenance*. Heraklion, Crete, Greece, pp. 291–298, 2011.

Stamatakis, A. RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics* 22 (21): 2688–2690, 2006.

Stamatakis, A. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics* 30 (9): 1312–1313, May, 2014.

Stein, L. D. Integrating biological databases. *Nature Reviews Genetics* 4 (5): 337–345, May, 2003.

Webber, J. A Programmatic Introduction to Neo4J. In *Annual Conference on Systems, Programming, and Applications: Software for Humanity (SPLASH)*. Tucson, Arizona, USA, pp. 217–218, 2012.

Woodman, S., Hiden, H., Watson, P., and Missier, P. Achieving Reproducibility by Combining Provenance with Service and Workflow Versioning. In *Workshop on Workflows in Support of Large-scale Science (WORKS)*. Seattle, Washington, USA, pp. 127–136, 2011.

Wozniak, J., Armstrong, T., Wilde, M., Katz, D., Lusk, E., and Foster, I. Swift/t: Large-scale application composition via distributed-memory dataflow processing. In *International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. Delft, Netherlands, pp. 95–102, 2013.

Zhao, J., Sun, F., Torniai, C., Bakshi, A., and Prasanna, V. A provenance-integration framework for distributed workflows in grid environments. In *Workshop on Grid and Utility Computing*. Cochin, India, pp. 17–20, 2008.

Zhao, Y., Wilde, M., and Foster, I. Applying the virtual data provenance model. In *International Provenance and Annotation Workshop (IPAW)*. Chicago, Illinois, USA, pp. 148–161, 2006.