# An Identity Crisis in the Life Sciences

Jun Zhao, Carole Goble, and Robert Stevens

School of Computer Science, University of Manchester, M13 9PL, U.K.
{zhaoj, carole, robert.stevens}@cs.man.ac.uk

**Abstract.** myGrid is an e-Science project assisting life scientists to build workflows that gather data from distributed, autonomous, replicated and heterogeneous resources. The provenance logs of workflow executions are recorded as RDF graphs. The log of one workflow run is used to trace the history of its execution process. However, by aggregating provenance logs of many workflow runs, one may gather the provenance of a common data product shared in multiple derivation paths. A successful aggregation relies on accurate and universal identification of each data product. The nature of bioinformatics data and services, however, makes this difficult. We describe the identity problem in bioinformatics data, and present a protocol for managing identity co-references and allocating identity to gathered and computed data products. The ability to overcome this problem means that the provenance of workflows in bioinformatics and other domains can be exploited to enhance the practice of e-Science.

## 1 Introduction

myGrid [1] is an e-Science project providing middleware services to assist bioinformaticians to perform *in silico* experiments [1]. myGrid uses workflows to orchestrate, access and interoperate a large number of public databases and applications, and manage those experiments and their *outcomes*, including data products, their provenance and experiment conclusions, using semantic-based metadata and data technologies [2]. Taverna, the workflow environment and workbench in myGrid, enables scientists to design and execute workflows, providing access to over 3,000 bio-resources. These services are mixtures of web services, grid services, java applications, database queries and scripts. Taverna has been used for gene alerting, gene and protein sequence annotation, proteomics, functional genomics, chemoinformatics, systems biology and protein structure prediction applications. Workflows have been used to identify a mutation associated with the autoimmune disorder Graves' Disease in the I kappa B-epsilon gene [3] and build the first complete and accurate map of the region of chromosome 7 involved in Williams-Beuren Syndrome (WBS) [4].

As part of the experiment design, scientists construct executable workflow definitions, written in Taverna's Scufl language [4], binding specific data, parameter settings and the end points of the services to be executed. Each execution of a workflow definition becomes a workflow run. Collections of workflow definitions

[1] http://www.mygrid.org.uk

L. Moreau and I. Foster (Eds.): IPAW 2006, LNCS 4145, pp. 254–269, 2006.

and workflow runs contribute to an overall experiment. myGrid collects both data produced during workflow runs and the provenance of these data products [2]. The provenance log of one workflow run is used to trace: the history of execution process, (e.g. services used); the origin of a data product, (e.g. the database or intermediate data product); and the ownership and intellectual property of each run (e.g. who and when). Ownership, purpose, etc are provenance annotations over experimental collections of workflow definitions as well as an individual workflow run. The provenance of each data product, gathered or computed, is automatically captured by detecting events during workflow enactment to form a dependency graph. Annotations of ownership and purpose are manual assertions of fact or opinion by the scientist on the data, processes or sub-graphs.

The workflows developed in the life sciences have a number of properties that impact on our provenance collection:

- The workflows are largely data pipelines, frequently generating data collections and then iterating in turn over each item in the collection.
- A workflow data product can be (a) a newly generated original data object, or (b) a pre-existing data object gathered from an external resource. Thus, when we refer to data products, these can be pre-existing objects that have been retrieved from an external collection (*gathered* data) or freshly computed data values (*computed* data).
- As public resources regularly change their content, the same workflow is rerun repeatedly over the same resources, perhaps with different parameter settings. The data products acquired by different runs are compared, merged and aggregated. A workflow can thus be executed repeatedly by the same user at a different time or location, or by different users from different research groups or institutions.
- The same data product may be acquired by different workflow runs under different experimental contexts, e.g. the user, the workflow definition, or the time of a run etc.

The final two points are important. Bioinformatics is an exploratory scientific discipline. Varying the settings of repeated executions might lead to completely different outcomes. Results and their provenance from repeated executions need to be accumulated to verify an ultimate conclusion. Thus, scientists need to put the provenance records to use: to aggregate, integrate and compare the provenance records for a common data product produced by multiple workflow runs.

**Definition 1.** *Consider a data product* $\mathsf{d}$ *acquired in a workflow run* $\mathsf{r}$*,*
- *the provenance log of* $\mathsf{r}$ *forms a graph* $\mathsf{P}(\mathsf{r})$*;*
- *the provenance log of* $\mathsf{d}$ *in* $\mathsf{r}$ *is a subgraph of* $\mathsf{P}(\mathsf{r})$*:* $\mathsf{P}(\mathsf{d},\mathsf{r})$*.*

Then, for $d$ acquired in both $r_1$ and $r_2$:

- Provenance *aggregation* for $d$ is to gather the provenance graphs $P(d, r_1)$ and $P(d, r_2)$.

- Provenance *integration* for $d$ is the merging of the aggregated provenance graphs: $P(d, r_1) \cup P(d, r_2)$ [5].
- Provenance pair-wise *comparison* for $d$ from runs $r_1$ and $r_2$ is the computed difference between the two provenance graphs: $P(d, r_1) - P(d, r_2)$ [6].

To aggregate, integrate and compare $P(d, r_i)$ $(i = 1, 2, ...n)$ for $d$ requires two things:

1. **a mechanism to merge and differ provenance graphs**: myGrid represents the workflow provenance metadata using technologies drawn from the semantic web community, chiefly the Resource Description Framework (RDF) [2]. RDF is essentially a simple graph data model. The RDF data store and query languages provide mechanisms for graph fusion and graph manipulation as well as querying.
2. **a mechanism to manage data object identity**: When we merge and differ provenance graphs, it is helpful if the same data object - a protein sequence, a gene, a database entry - has the same identity regardless of its origin. RDF provides an explicit identification system, Universal Resource Identifiers (URIs), for identifying resources to allow metadata about a resource to be merged from several sources. We identify gathered data objects, computed data products, workflows, parameters, etc by allocating LSIDs (Life Science Identifiers) [7] to them.

Representing provenance using RDF and LSIDs enables us to potentially aggregate multiple provenance graphs, $P(d, r_i)$ $(i = 1, 2, ...n)$. Although the RDF-based graph model and associated manipulation and query mechanisms lend themselves to the support of cross-run or cross-workflow provenance aggregation and integration, the allocation and management of identity is problematic. LSIDs are proposed as global unique identifiers (GUIDs) by the life science community [7,8]. This scheme has been applied to major life science databases, such as NCBI [3], UniProt [4], and Affymetrix [5]. In many e-Science domains, such as chemistry, physics, astronomy, etc, GUIDs for data objects are taken for granted. An example is the Digital Object Identifier (DOI) [9] for digital publications.

However, multiple identities are allocated for the same data object in life sciences. The identity allocation scheme in Taverna does not guarantee a universal identity to be allocated for *equivalent* data (defined in Section 2) produced in multiple runs. These multiple identities for the same or equivalent data are called *polyonomous* identities [6], which lead to an identity crisis in inter-run provenance aggregation, integration and comparison.

The rest of this paper is organized as follows: Section 2 describes the motivation for managing data identities and presents a simple, real workflow. Section 3 highlights the life science identity landscape, showing how the difficulties in

[2] http://www.w3.org/RDF/
[3] http://www.ncbi.nlm.nih.gov/Genbank/
[4] http://www.ebi.uniprot.org/
[5] http://lsid.biopathways.org/authorities.shtml
[6] http://dictionary.reference.com/search?q=Polyonomous

allocating identities to data products in a coherent fashion have consequences on the recording and aggregating of provenance records. In Section 4, we propose a new identity protocol to construct identity co-references and the introduction of a new identity naming scheme. We show how identity co-references can help when comparing provenance generated in repeated runs of the presented real workflow. Related identity work and identity management in provenance are described in Section 5. We conclude with a discussion and summary of the characteristics of the identity problem.

## 2 Collecting Provenance from a Taverna Workflow

Figure 1 schematically presents a simple workflow (`WF1`) from the WBS study. This workflow identifies a collection of DNA sequences from a database, similar to the query sequence. Step 1 invokes the BLAST (Basic Local Alignment Search Tool) service [10] using the initial query sequence and a set of configuration parameters. BLAST detects regions of similarity embedded in otherwise unrelated proteins or nucleic acids. Step 2 simplifies the BLAST report data product and extracts the DNA sequences contained from this report. Step 3 retrieves the GenBank report for each DNA sequence produced in step 2 and produces a collection of GenBank reports.

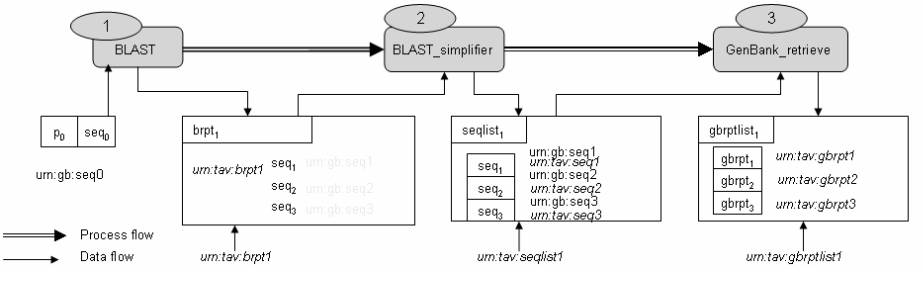


**Fig. 1.** Workflow (`WF1`) which forms part of the WBS case study. The BLAST report $brpt_1$, the sequence list $seqlist_1$, the GenBank report list $gbrptlist_1$ and each GenBank report $gbrpt_i$ are computed data products. Each sequence ($seq_i$) pre-exists, gathered and processed by the workflow. During the execution of the workflow, each $seq_i$ has an external identity (e.g. `urn:gb:seq1`). This identity is lost as it becomes a text string by step 1, which is subsequently recovered by step 2 and associated with an additional Taverna identity.

The bottom of the figure shows the data products consumed and produced in one run of `WF1`.

- In step 1, the input to the BLAST service is a DNA sequence ($seq_0$) to align against, and a set of parameter settings $p_0$ for invoking the service, which includes: the $database_0$ of sequences, the statistical significance threshold $evalue_0$ for reporting sequence matches, and the maximum number of reported $scores_0$ in the BLAST report. The output of step 1 is a BLAST

report, $brpt_1$, containing a collection of sequence data entries retrieved from $database_0$: $Contains(brpt_1) = \{seq_i, 1 \leq i \leq n\}$. The sequences and their identifiers are embedded as text in the report, along with other materials.
- In step 2, the $brpt_1$ produced in step 1 is parsed and simplified by the `BLAST_Simplifier` service. This service extracts the sequence entries in $brpt_1$ to recover a collection of DNA sequence data objects: $seqlist_1 = \{seq_i, 1 \leq i \leq n\}$.
- In step 3, a GenBank report $gbrpt_i$ is retrieved by the `GenBank_Retrieve` service for each sequence in $seqlist_1$ produced in step 2. The output of step 3 is a collection of GenBank reports: $gbrtplist_1 = \{gbrpt_i, 1 \leq i \leq n\}$.

### 2.1 Gathered and Computed Data

A data product can be either *computed* or *gathered*, which decides the identities it might be associated with. A data product can be either *atomic* or a *collection*, which decides the provenance metadata captured for it in myGrid.

- A computed data product is generated as a consequence of a workflow execution. The BLAST report $brpt_1$, the GenBank report $gbrpt_1$, the collection of sequences $seqlist_1$ and the collection of GenBank reports $gbrptlist_1$ are computed data products.
- A gathered data product is one that was pre-existing and has been retrieved from external databases. Each protein sequence entry $seq_i$ from the sequence database, contained in the $brpt_1$ is a gathered data product.

A computed or gathered data product can be either an atomic data product or a collection data product. Each collection data product contains a collection of elements, which are either atomic or collection data products, following the usual recursive composite pattern. Whether a data product is atomic or a collection can be rather subtle, dependent on the domain view or the transportation view:

- A domain collection is classified by its data content, for example, $brpt_1$ is collection data product at the domain level, containing a collection of gathered sequences $\{seq_i\}$. Every $seq_i$ is an atomic data product.
- A transportation collection is decided by whether the data product is treated as a single data product or as a list of data products in Taverna. For example, $brpt_1$ is an atomic data product at the transport level when it is transferred between services in the workflow runs. The $seqlist_1$ and the $gbrptlist_1$ in Figure 1 are collection data products, as they are transferred as *lists* between services.

### 2.2 Using Provenance Graphs

In myGrid, data provenance gathered in each workflow run forms a graph, shown in Figure 2, with data products as the nodes and their *provenance* relationships as the edges. Consider a data product $d_i$ produced in a run, either an atomic or a collection, there are two types of provenance relationships recorded in the myGrid provenance:

– *derivedFrom*: $d_i$ is derived from another data product $d_j$, or $d_i$ is derived from a set of parameter settings $p_j$. For example, the BLAST report $brpt_1$ is derived from the input sequence $seq_0$ and from the set of parameters $p_0 = \{database_0, evalue_0, score_0\}$.
– *elementOf*: $d_i$ could be an element of a collection data product $d_j$ ($d_i \neq d_j$). For a $seq_i$ of $seqlist_1$, atomic $seq_i$ is an element of the collection data product $seqlist_1$. The myGrid provenance model only captures the relationship between a transportation collection and its elements.

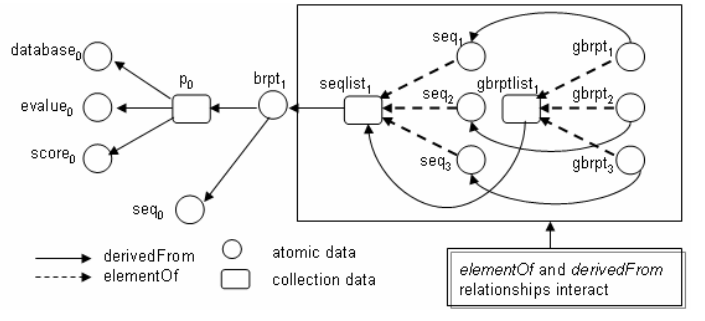


**Fig. 2.** Data provenance graph formed in one run of of `WF1`

A bioinformatics workflow is often repeated over the same resources or varying settings. For example, considering the WF1, one might use:

1. the same BLAST service and same parameter settings to obtain updated data products. As the public databases are frequently updated, WF1 needs to be frequently repeated in order to collect updated sequences.
2. the same BLAST service but different parameter settings to look for the best parameter settings for this workflow.
3. different BLAST services, such as DDBJ BLAST[7], WU-BLAST[8], to look for the best service for this workflow or to verify that consistent results can be obtained with varying BLAST services.

The data products of these repeated runs of `WF1` are compared to reach conclusions, e.g. an updated sequence was produced in a rerun of `WF1`, or "`score=100`" is the best parameter setting for running `WF1`. These conclusions need to be justified using the provenance of experiment data products.

In repeated workflow runs, multiple data provenance graphs are produced, which contain some similarities and differences. The same data object can be gathered in different runs; e.g. the protein sequence $seq_1$ appears as a result in both $r_1$ and $r_2$ of `WF1`. Thus both provenance graphs $P(seq_1, r_1)$ and $P(seq_1, r_2)$ have $seq_1$ in *common*. Data objects that contain the same data values can be

[7] `www.xml.nig.ac.jp/wsdl/index.jsp`
[8] `http://blast.wustl.edu/`

computed in different runs, e.g. $brpt_1$ from $r_1$ and $brpt_2$ from $r_2$ contain exactly the same sequences, despite changes in time, parameters or the contents of the external database. Thus $brpt_1$ and $brpt_2$ in the two graphs $P(brpt_1, r_1)$ and $P(brpt_2, r_2)$ *correspond*. In this paper *equivalent* data products include both gathered products in common and corresponding products that are computed.

In order to *aggregate* the provenance of a $d$ computed or gathered in multiple runs, we need to identify this $d$ produced in each run. In order to *integrate* and *compare* the multiple provenance graphs of $d$, i.e. $P(d, r_i)$ $(i = 1, 2, ...n)$, we need to identify all the equivalent data products and parameter settings recorded in these graphs. The provenance graph in Figure 2 is represented by RDF in myGrid. Each data product and control parameter in this graph is identified by a URI. We need to manage the identities to make sure that equivalent data products and parameters are identified uniquely and universally across workflow runs. In the next section we explain the identity issues for `WF1`.

## 3 Identities

In Figure 1, all identities of all data products by a workflow are managed by the LSID protocol. An LSID consists of five parts separated by colons: a prefix (`urn:lsid`); the authority name (`www.mygrid.org.uk`); the authority-specific data namespace (`data`); the namespace-specific object identifier (`49841`) and a version number of the object (`1`) leading to a URI: `urn:lsid:www.mygrid.org.uk:data:49841:1`. An LSID authority for a resource allocates an identity and resolves it for the resources for which it is an authority (and no others), guaranteeing that the data is immutable. Each data provider has a responsibility for managing its own LSID authority. Even when resources are replicated locally a different, local LSID authority is in place.

A *gathered* data product may carry an external identity, but a *computed* data product is assigned with a Taverna identity. This impacts on the protocol for managing data identities as shown in Section 4. Using the workflow `WF1` in Figure 1, we now explore external identity allocation by data resources and internal identity allocation by Taverna.

### 3.1 External Identity: Resource Generated Identities

At least 700 different, heterogeneous resources are available in life sciences [11]. The autonomy of data providers enables rapid generation and deployment of new resources, but they rarely conform to any community-wide standards, often allocating different identities for a common data object. We find the following situations:

– equivalent data objects in different databases. For example, "`gi:15145617`" (GenBank) and "`ac073846`" (EMBL-Bank[9]) are the same DNA sequence. The protein "Dual specificity DE phosphatase Cdc25C" is identified as "`aaa35666`" in GenBank and "`p30307`" in UniProt.

[9] `http://www.ebi.ac.uk/embl/`

– equivalent data objects in different replicas, a variation of the previous point. A resource is often replicated remotely or locally, and sometimes locally extended or customised. For example, the same sequence is "`urn:lsid:myg:ac073846`" for a local copy of EMBL-Bank in myGrid with its own LSID authority.
– equivalent data objects from different workflows. The invocation of a Kyoto Encyclopedia of Genes and Genomes (KEGG[10]) pathway service (instead of a BLAST service) produces a pathway result containing a collection of protein sequences. Some of the sequences in the $brpt_1$ in Figure 1, also appear in the pathway data product, but now with KEGG LSIDs.

These arrangements result in polyonomous external identities for equivalent data products gathered from different databases, replicas or by different services.

### 3.2 Taverna Identity: Workflow Generated Identities

Figure 3 gives the myGrid LSID allocation architecture. In a workflow run, when a data product is passed to the enactor, it is allocated a Taverna LSID by the Taverna LSID authority. This identity is associated with the data product when it is stored or passed to invoked other services by the enactor. The data products acquired (gathered or computed) by a run are stored in a customized database as part of the workflow or a local "catch all" store, the relational data store BACLAVA. This identity is also used to store the RDF provenance metadata of this data product in the metadata store, KAVE. Data and provenance are immutable in myGrid. Once data products and provenance are preserved, they should not be deleted or altered. Provenance of a data product can be augmented with more provenance metadata. A client communicates the myGrid data and provenance repositories over the network by the LSID protocol [8] to retrieve data or metadata of a data product by its LSID.

**Migration Polyonomy.** Data products generated by Taverna and stored in a database or the BACLAVA store can be archived or replicated among scientists in their own file stores. A migration polyonomy is caused by the failure to migrate data identities with the data when the data are curated. For instance, when a data product is copied from the BACLAVA store to a personal file system, its identity is deprecated and replaced with a new identity such as a path to access the file system.

**Execution Polyonomy.** In each independent run, the Taverna LSID authority is ignorant of the existence of common or corresponding data products produced in different runs and the existence of the polyonomous identities allocated for these data products. Therefore, polyonomous identities for common or corresponding data products are produced by repeated executions:

1. *corresponding computed data products.* Two corresponding $brpt_1$ and $brpt_2$ generated in two runs of `WF1` contain exactly the same protein sequence

[10] `http://www.genome.jp/kegg/`

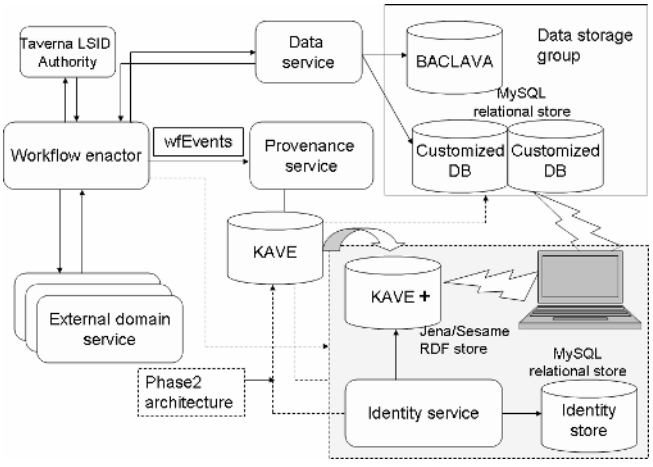


**Fig. 3.** myGrid's architecture for allocating LSIDs for data products during workflow executions. The dashed part represents the extended architecture when introducing the identity service to manage the identities for common or corresponding data products.

objects. $brpt_1$ and $brpt_2$ are identified by different Taverna identities as they are from different runs and are, in fact, different reports.

2. *common gathered data products.* Let $d_1$ be the same $seq_1$ gathered in two runs $r_1$ and $r_2$, two provenance graphs $P(d_1, r_1)$ and $P(d_1, r_2)$ share this common $d_1$ (see Figure 4(a)). This $d_1$ should be given the same identifier in order to assert that $d_1$ in $r_1$ and $r_2$ are equivalent. However, the processing of the workflows means that this is not the case.
   - In Figure 1, when step 1 is finished, the computed data product $brpt_1$ is stored and allocated a Taverna LSID `urn:tav:b1`. The gathered data objects, i.e. the $\{seq_i\}$ contained within $brpt_1$ are neither extracted nor allocated with any LSIDs. This is because they have been turned into text by BLAST and their external identities, e.g. `urn:gb:seq1`, are contained in their data contents as strings.
   - When step 2 is finished, each sequence, such as $seq_1$, has been extracted and stored. Because it is a new object, recovered by post-processing, it is automatically allocated a Taverna LSID `urn:tav:seq1`, despite the fact it already has an external LSID that it carried. In each workflow run of `WF1`, if $seq_1$ appears it is given a new, different Taverna LSID. Thus the same data product has its external LSID and, for each run, a Taverna LSID.

Two cases further compound the problem:

1. corresponding computed collection data products at the transportation level. For instance, the data product from the `GenBank_retrieve` service $gbrptlist_1$ contains a collection of GenBank reports $\{gbrpt_i\}$. This collection data prod-

uct and its elements are always allocated new, different Taverna identities each time they are produced.
2. equivalent nested data objects in a data product. For instance, each sequence $seq_i$ contains some nested data objects such as the species data object. These nested data objects are allocated with new, different Taverna identities each time they are extracted.

Polyonomies are not only due to inadequate attention to our identity allocation mechanism, but are inevitable. Firstly, we need to differentiate the data product and its data derivation path produced in one context with its equivalent data product produced in another context. Figure 4(a) shows the two derivation paths for the data product $d_1$ that were gathered in different runs. If during the provenance collection this $d_1$ was identified by the same identity, as shown in Figure 4(b), only the merged derivation path will be kept in KAVE. It becomes difficult to retrieve "the data product which $d_1$ was derived from that was produced in run $r_1$". myGrid tries to solve this problem by incorporating more context information with a data product using named graphs [12], as discussed in Section 5.

Secondly there are potential computation costs of avoiding publishing polyonomous identities at run time for equivalent data products. The equivalence between computed data products is based on their values most of the time, which is inefficient. For instance, the identity correspondence of our example BLAST reports cannot be decided based on the identities of the sequence data objects contained in these data products, as polyonomous identities are published by different databases for the same sequence. Evaluating the equivalence of data products at run time could slow down the workflow enactor, but is achievable by a post workflow enactment process.

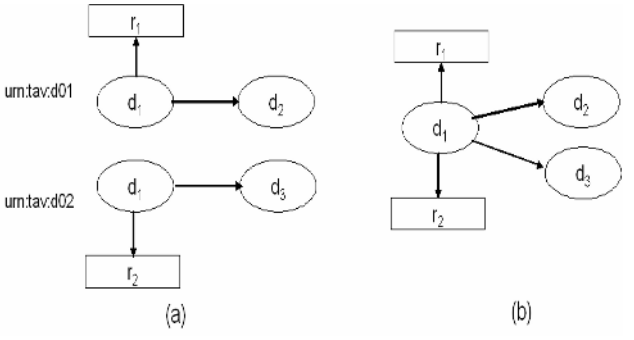


**Fig. 4.** (a) $d_1$ identified as `urn:tav:d01` was derived from $d_2$ in the run $r_1$ and its equivalent data product $d_1$ identified as `urn:tav:d02` was derived from a different data product $d_3$ in the run $r_2$. (b) If $d_1$ is identified by one identity, the data derivation paths for $d_1$ that were generated in different runs will be converged.

## 4 Identity Solutions

In order to cope with the *execution* identity problem we propose an identity protocol for building co-references of execution polyonomies. This protocol is

asynchronous to workflow enactment and data and provenance collection. We ignore external polyonomies in this protocol.

### 4.1 Co-reference Identity Protocol

**Definition 2.** *Consider a data product* $\mathsf{d}$*, the* $\mathsf{IDSet(d)} = \{\mathsf{id_1}, \mathsf{id_2}, ..., \mathsf{id_n}\}$ *stores a set of polyonomous identities for* $\mathsf{d}$*. Each* $\mathsf{IDSet(d)}$*,*

- *has its own identity:* $\mathsf{id(IDSet(d))}$*;*
- *is associated with the content of* $\mathsf{d}$*.*

For example we have $IDSet(brpt_1) = \{\texttt{urn:tav:b0}\}$ for $brpt_1$. IDSet objects are stored in the identity store by the identity service in Figure 3. The dashed part in Figure 3 shows the architecture of how the identity protocol functions when executing a workflow in Taverna and interacts with other components in $^{my}$Grid. Three actors participate in the identity protocol: the external service, the workflow enactor and an identity service.

1. The workflow enactor passes data to external services, then invokes these services based on the workflow definition.
2. The service returns the data results and the external identities for gathered data products. When the service returns gathered data products in a response message to the enactor, it should publish identities for these data products in its response message. If the service failed to do this, it would be regarded as a service with lower quality than those that do so.
3. When the data products are returned to the enactor by the service, the enactor assigns Taverna LSIDs to these data products.
4. The enactor invokes the identity service, passing messages to the identity service that include: the data products, their Taverna LSIDs and associated external LSIDs (if any).
5. The identity service intercepts Taverna identities for gathered and computed data products and retrieves or builds IDSet objects for these data. It updates the IDSet objects in the identity store and inserts these IDSet identities as metadata of the data products into the provenance store KAVE.The enactor stores the data products in the data store BACLAVA and their provenance in the provenance store KAVE.

The step of building IDSet objects can be taken offline, by analyzing the provenance store and the data store after the workflow run. A relational identity store is used for keeping the IDSet objects. The RDF KAVE store contains provenance metadata as well as the IDSet metadata for the data products. We name this upgraded KAVE as *KAVE+*. These IDSet metadata are generated by the following scheme and can be used when integrating and comparing provenance graphs, as shown in the Section 4.3.

### 4.2 Revised Identity Naming Scheme

This identity protocol constructs and updates IDSet objects by three means: (1) identities of gathered data products, (2) data values of computed data products and (3) data objects contained in collection data products at the transport level.

**Allocation 1. Allocation by Identities.** The identity service receives a *gathered* data product $d$, either atomic or collection:

1. Search for an existing IDSet object using the external identity of $d$. If an IDSet is found, retrieve it; otherwise create one.
2. Update the identity store with $d$'s external and Taverna identities, and its IDSet object identity, $id(IDSet(d))$.
3. Insert this IDSet object and its relationship with $d$ as RDF statements into the provenance metadata store KAVE+.

**Allocation 2. Allocation by Values.** The identity service receives a *computed* data product $d$, either an atomic or a collection, e.g. a *brpt*, which collects a set of external data objects:

1. Search for an existing IDSet object using $d$'s data value. If an IDSet is found, retrieve it; otherwise create one.
2. Update the identity store with the Taverna identity of $d$ and the IDSet object identity, $id(IDSet(d))$.
3. Insert this IDSet object and its relationship with $d$ as RDF statements into the provenance.

**Allocation 3. Allocation by Objects.** The identity service receives a *transportation collection* data product $d$, which contains a collection of atomic or collection data products. This $d$ and each of its elements is identified by a Taverna LSID, e.g. the $seqlist_1$ containing a collection of atomic gathered data products $seq_i$ in Figure 1.

**Definition 3.** *Consider two collection data products* $\mathsf{d_i}$ *and* $\mathsf{d_j}$ $(\mathsf{d_i} \neq \mathsf{d_j})$, $\mathsf{d_i}$ *and* $\mathsf{d_j}$ *are equivalent,* $\mathsf{d_i} \equiv \mathsf{d_j}$*, if and only if both* $\mathsf{d_i}$ *and* $\mathsf{d_j}$ *have the same size, and all elements in them are equal.*

For a collection data product $d$, the identity service should:

1. Search for any existing IDSet objects for each element of $d$. Search by the element's identity if it is a gathered data product; and search by the element's value if it is a computed data product.
2. Search for $d$'s equivalent collection data product and an existing IDSet object. If an IDSet is found, retrieve it; otherwise create one.
3. Update the identity store with $d$'s Taverna identity, and its IDSet object identity, $id(IDSet(d))$.
4. Insert the relationship of $d$ and its element data products into the identity store.
5. Insert this IDSet object and its relationship with the data product $d$ as RDF statements into the provenance metadata store.

This protocol repairs the execution polyonomies in Taverna. Currently, polyonomous identities continue to be allocated in $^{\text{my}}$Grid to avoid the costs of allocating unique identities for equivalent data products during workflow runs. This protocol and naming scheme, however, improves the maintenance of: (a)

the external identities associated with gathered data products; (b) the relationship between computed data product and its element data products, computed or gathered; and (c) Taverna identity co-references.

### 4.3 Putting the Identity Service to Use

If equivalent data products are identified by the same identity, the provenance aggregation, integration and comparison will be possible. The IDSet resolves this problem by maintaining a collection of polyonomous identities for a $d$. The identity of an IDSet object provides a universal identity for a $d$ in myGrid. An identity service prototype was implemented as a plug-in to Taverna, building IDSet objects for equivalent data products during workflow runs:

- For a `gathered` data product, such as the protein sequence $seq_1$, its IDSet object is built by the data product's external identity.
- For a `computed` data product, such as $brpt_1$, the external identities of its elements are parsed and extracted from the data content. $brpt_1$'s IDSet object is built by these external identities of its elements.
- For a `collection` data product at the transportation level, such as the $seqlist_1$, $gbrptlist_1$ in Figure 1, its IDSet object is built by the IDSet identities of its element data products.

To show how the identity service can help us achieve the goal of integrating provenance graphs from repeated runs of every pair-wise runs $r_1$ and $r_2$ of `WF1`, we conduct the following:

1. Aggregate the data provenance graphs. This returns two provenance graphs $P(r_1)$ and $P(r_2)$.
2. Normalize each provenance graph $P(r_i)$. For each data product in $P(r_i)$, retrieve its IDSet identity. If an IDSet identity is found, replace the data identity in $P(r_i)$ with the IDSet identity. This operation returns a normalized graph, $P^n(r_i)$, with each equivalent data product produced in different runs being identified by its IDSet identity.
3. Compare the normalized graph $P^n(r_1)$ and $P^n(r_2)$.

We succeeded in detecting the similarities and differences between $P(r_1)$ and $P(r_2)$. This approach is much faster than identifying equivalent data products at the time of comparison. The number of data objects contained in each provenance graph $P(r_i)$ $(i = 1, 2...n)$ decides the size of $P(r_i)$ and impacts the computational complexity of normalizing a $P(r_i)$. An optimization of this normalization is needed if a large provenance graph is to be processed.

## 5 Related Work

Polyonomous identities are a common problem in data integration. RefSeq [13] builds cross-references for sequence data across multiple major sequence databases. The Handle system provides GUIDs for digital objects assured by a global

naming authority [14]. This is hard to achieve in life sciences because of the autonomy of data and tool providers. Our identity protocol focuses on constructing co-references between Taverna polyonomies for equivalent data products. These identity co-references can be published at different places and then merged by set calculus. This identity protocol can be adopted in many service-oriented architectures such as the provenance collection architecture proposed by Groth [15].

The Virtual Data Language (VDL) [16] represents derivation relationships between data that were processed by computational procedures. It is unclear whether the aggregation or integration of provenance over repeated reruns of the same workflow over the same data is an issue. The focus is on computed data products and the identity issue becomes one of data mining for identical data values rather than the maintenance of external data object identity and taking care not to allocate polyonomous identities.

In scientific study it is important to harvest provenance logs across runs, initialized by different users or groups [17]. But no naming scheme is defined in up to date provenance work to avoid polyonomous identities for equivalent data products. VisTrails traces the provenance of how workflows are revised from one to another [18]. However, the identity problem for data products remain un-clarified.

If unique identities are allocated for equivalent data products, a mechanism is required to differentiate the different contexts in which each data product are produced. The PASOA project includes a workflow run id as part of the data product id that is produced in a particular workflow run [15]. However, additional contextual information is needed by bioinformaticians, such as the person who produced the data, and an intermediate data product from which a collection of data products were derived from. myGrid adopts named graphs [12] to incorporate more contextual information with data products.

## 6 Conclusion

In previous work we focused on building a provenance model and the technology to represent this model [2]. The model was developed to assist not only in keeping audit trails of a single workflow run, but also to support the analysis of multiple provenance logs across multiple workflow runs. Although all the provenance graphs are represented as RDF graphs, analyzing these graphs requires a scheme to identify equivalent data products; both common gathered products and corresponding computed products. Allocating and assigning data product identity proved harder than anticipated in practice for provenance graphs that are independently generated, yet need to be combined. Our previous identity allocation scheme works well when we do not aggregate, integrate and compare provenance, but raises issues when we do.

Bioinformatics workflows not only generate new data, but also discover new information by combining and collating existing data. These pre-existing data, external to the Taverna world, have their own identities allocated. In addition, local identities are published for these data products, such as Taverna LSIDs and

VDL's Logical File Names. The autonomous nature of the current bioinformatics domain makes it hard to adopt a global naming service unless a community-wide agreement is achieved, such as the data transfer standard in earth science [19] and in the caBIG project [11]. Our problem is particularly acute in that we do not prescribe a closed, strongly typed data environment such as that dictated by caBIG. The multiple identity problems are present in a large extent such as on the Web. Polyonomous identities point to the same entity hosted by different web sites. In this paper we tried to enumerate the different identity duplication problems for equivalent data products. To resolve this problem we revise our identity allocation scheme and construct co-references between polyonomous identities.

In this paper we aggregate, integrate and compare provenance graphs from repeated runs of the same workflow. We aim also to analyze provenance graphs from runs of *corresponding* workflows. Workflows using different services in the workflow definitions, but realizing the same experiment goal and function correspond. Corresponding workflows evolved one from another. To analyze provenance graphs from runs of corresponding workflows requires a full-fledged definition of the corresponding relationship among workflows and an infrastructure to maintain this relationship, as presented in [18].

Two scalability issues remain to be solved: (1) the scalability of identifying corresponding collection data products computed in iterations in different runs. Iterations over iterations in a workflow run produce collections with multiple hierarchies, i.e. collections containing collections iteratively. This makes it expensive to retrieve corresponding collections with multiple hierarchies; and (2) the scalability of exploiting identities. As shown by the example of exploiting identities in Section 4.3, the computation complexity of normalizing a provenance graph is decided by the size of the provenance graph. This normalization is required in provenance integration and comparison. An optimization is needed if a large provenance graph is to be normalized.

## Acknowledgements

The myGrid project, grant numbers GR/R67743, EP/D044324/1 and EP/C536444/1, is funded under the UK e-Science programme by the EPSRC. The authors would like to acknowledge the other members of the myGrid team for their contributions, and in particular acknowledge Tom Oinn, Matthew Pocock, Daniele Turi and Chris Wroe. We thank Stian Soiland and David Withers for their useful comments.

## References

1. Stevens, R., Tipney, H.J., Wroe, C., Oinn, T., Senger, M., Lord, P., Goble, C., Brass, A., Tassabehji, M.: Exploring Williams-Beuren Syndrome Using myGrid. Bioinformatics **20** (2004) 303–310

[11] https://cabig.nci.nih.gov/

2. Zhao, J., Wroe, C., Goble, C., Stevens, R., Quan, D., Greenwood, M.: Using semantic web technologies for representing e-science provenance. In: Proc. of the Third International Semantic Web Conference. Volume 3298. (2004) 92–106
3. Li, P., Hayward, K., Jennings, C., Owen, K., Oinn, T., Stevens, R., Pearce, S., Wipat, A.: Association of variations on i kappa b-epsilon with graves' disease using classical and mygrid methodologies. In: Proc. of the UK e-Science AHM. (2004)
4. Oinn, T., Greenwood, M., Addis, M., Ferris, J., Glover, K., Goble, C., Hull, D., Marvin, D., Li, P., Lord, P., Pocock, M.R., Senger, M., Wipat, A., Wroe, C.: Taverna: Lessons in creating a workflow environment for the life sciences. Journal of Concurrency and Computation: Practice and Experience (2005) in press.
5. Weisstein., E.W.: (Graph union) `http://mathworld.wolfram.com/GraphUnion.html`.
6. Weisstein., E.W.: (Graph difference) `http://mathworld.wolfram.com/GraphDifference.html`.
7. Clark, T., Martin, S., Liefeld, T.: Globally distributed object identification for biological knowledgebases. Briefings in Bioinformatics **5** (2004) 59–70
8. Martin, S., Hohman, M.M., Liefeld, T.: The impact of life science identifier on informatics data. Drug Discovovery Today. **10** (2005) 1566–72
9. Dalziel, J.: DOI in a DRM environment. White paper, Macquarie University (2004)
10. Altschul, S., Gish, W., Miller, M., Myers, E., Lipman, D.: Basic local alignment search tool. Journal of Molecular Biology **215** (1990) 403–410
11. Galperin, M.Y.: The molecular biology database collection: 2006 update. Nucl. Acids Res. **34** (2006) 3–5
12. Carroll, J., Bizer, C., Hayes, P., Stickler, P.: Named graphs. Journal of Web Semantics **3** (2005)
13. Pruitt, K.D., Maglott, D.R.: Refseq and locuslink: Ncbi gene-centered resources. Nucleic Acids Research **29** (2001) 137–140
14. Kahn, R., Wilensky, R.: A framework for distributed digital object services. Technical Report tn95-01, Macquarie University (1995)
15. Groth, P.T., Luck, M., Moreau, L.: A protocol for recording provenance in service-oriented grids. In: Proc. of the Eighth International Conference on Principles of Distributed Systems, Grenoble, France (2004) 124–139
16. Foster, I., Vockler, J., Wilde, M., Zhao, Y.: The virtual data grid: A new model and architecture for data-intensive collaboration. In: Proc. of the First Biennal Conference on Innovative Data System Research. (2003)
17. Futrelle, J.: Harvesting rdf triples. In: Proc. of the Third International Provenance and Annotation Workshops. (2006) in press.
18. Bavoil, L., Callahan, S.P., Crossno, P.J., Freire, J., Scheidegger, C.E., Silva, C.T., Vo, H.T.: Vistrails: Enabling interactive multiple-view visualizations. In: Proc. of IEEE Visualization. (2005) 135– 142
19. Arctur, D.K., Hair, D., Timson, G., Martin, E.P., Fegeas, R.: Issues and prospects for the next generation of the spatial data transfer standard (SDTS). International Journal of Geographical Information Science **12** (1998) 403 – 425