COMPUTING SCIENCE

D-PROV: extending the PROV provenance model with workflow structure

Paolo Missier, Saumen Dey, Khalid Belhajjame, Victor Cuevas-Vicenttin and Bertram Ludaescher

TECHNICAL REPORT SERIES

No. CS-TR-1375 **February 2013**

TECHNICAL REPORT SERIES

No. CS-TR-1375 February, 2013

D-PROV: extending the PROV provenance model with workflow structure

Paolo Missier, Saumen Dey, Khalid Belhajjame, Victor Cuevas-Vicenttin, Bertram Ludaescher

Abstract

This paper presents an extension to the W3C PROV provenance model, aimed at representing process structure. Although the modelling of process structure is out of the scope of the PROV specification, it is beneficial when capturing and analysing the provenance of data that is produced by programs or other formally encoded processes. In the paper, we motivate the need for such extended model in the context of an on-going large data preservation project, DataONE, where provenance traces of scientific workflow runs are captured and stored alongside the data products. We introduce new provenance relations for modelling process structure along with their usage patterns, and present sample queries that demonstrate their benefit.

Printed and published by Newcastle University,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

MISSIER, P., DEY, S., BELHAJJAME, J., CUEVAS-VICENTTIN, J., LUDAESCHER, B.

D-PROV: extending the PROV provenance model with workflow structure
[By] Paolo Missier, Saumen Dey, Khalid Belhajjame, Victor Cuevas-Vicenttin, Bertram Ludaescher

Newcastle upon Tyne: Newcastle University: Computing Science, 2013.

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1375)

Added entries

NEWCASTLE UNIVERSITY
Computing Science. Technical Report Series. CS-TR-1375

Abstract

This paper presents an extension to the W3C PROV provenance model, aimed at representing process structure. Although the modelling of process structure is out of the scope of the PROV specification, it is beneficial when capturing and analysing the provenance of data that is produced by programs or other formally encoded processes. In the paper, we motivate the need for such extended model in the context of an on-going large data preservation project, DataONE, where provenance traces of scientific workflow runs are captured and stored alongside the data products. We introduce new provenance relations for modelling process structure along with their usage patterns, and present sample queries that demonstrate their benefit.

About the authors

Dr. Paolo Missier is a Lecturer in Information and Knowledge Management with the School of Computing Science, Newcastle University, UK. His current research interests include models and architectures for the management and exploitation of data provenance, specifically extensions of e-infrastructures for scientific provenance. He is the PI of the EPSRC-funded project "Trusted Dynamic Coalitions", investigating provenance-based policies for information exchanges amongst partners in the presence of limited trust. Paolo contributes to two Provenance-focused Working Groups: he is a member of the W3C Working Group on Provenance on the Web, where co-edited the Provenance Conceptual Data Model; and he is co-lead of the Provenance Working Group of the NSF-funded DataONE project. He serves on a voluntary basis in both capacities. Paolo holds a Ph.D. in Computer Science from the University of Manchester, UK (2007), a M.Sc. in Computer Science from University of Houston, Tx., USA (1993) and a B.Sc. and M.Sc. in Computer Science from Universita' di Udine, Italy (1990).

Saumen Dey is a Phd student in the computing science department at UC Davis, USA.

Khalid Belhajjame is a researcher at the school of computer science of the University of Manchester. He obtained his PhD from the University of Grenoble in 2004 under the supervision of Prof. Christine Collet and Dr. Genoveva Vargas Solar. His research interests include semantic annotations of web services, scientific workflows, dataspaces, and data mapping and integration.

Victor Cuevas-Vicenttin is a postdoctoral researcher in computer science at the University of New Mexico and the University of California at Davis, working for the DataONE Project.

Bertram Ludäscher is a Professor of Computer Science at the Department of Computer Science and the Genome Center at the University of California, Davis. Prior to joining UC Davis, he worked at the San Diego Supercomputer Center at UC San Diego where until 2004 he was an Associate Research Scientist, leading the Knowledge-Based Information Systems lab. Dr. Ludäscher's primary research interests are in scientific data management, in particular scientific data integration, scientific workflow management, and knowledge-based (semantic) extensions thereof. He is also interested in foundations of databases, e.g., query languages and query rewriting. He received his M.S. in Computer Science (*Dipl.-Inform.*) from the Technical University of Karlsruhe in 1992, and his Ph.D. (*Dr.rer.nat.*) in Computer Science from the University of Freiburg in 1998, both in Germany.

Suggested keywords

PROVENANCE
WORKFLOW

D-PROV: Extending the PROV Provenance Model with Workflow Structure

Paolo Missier
Newcastle University, UK

Saumen Dey
UC Davis, CA, USA

Khalid Belhajjame
University of Manchester, UK

Víctor Cuevas-Vicenttín
UC Davis, CA, USA

Bertram Ludäscher
UC Davis, CA, USA

Abstract

This paper presents an extension to the W3C PROV[1] provenance model, aimed at representing process structure. Although the modelling of process structure is out of the scope of the PROV specification, it is beneficial when capturing and analyzing the provenance of data that is produced by programs or other formally encoded processes. In the paper, we motivate the need for such and extended model in the context of an ongoing large data federation and preservation project, DataONE[2], where provenance traces of scientific workflow runs are captured and stored alongside the data products. We introduce new provenance relations for modelling process structure along with their usage patterns, and present sample queries that demonstrate their benefit.

1 Introduction

The Provenance Interchange Working Group has recently released PROV [17], a W3C recommendation. PROV was developed with the aim of promoting interoperable interchange of provenance information in heterogeneous environments such as the Web. It is defined using an abstract relational model as well as an OWL ontology, with multiple serializations, including RDF and XML. PROV is generic and domain-independent, as it does not cater for the specific requirements of specific systems or domain applications. Rather, it provides extension points through which such systems and applications can extend PROV for their purposes. In this paper we are concerned with the modelling of provenance traces generated by the execution of scientific workflows. We begin by motivating the need, in this setting, to extend and complement the trace of a workflow execution with a representation of the workflow itself. We note that PROV alone does not accommodate such additional modelling features, present D-PROV, an extension to PROV, and show that it fulfills these needs.

This work is set in the context of DataONE (Data Observation Network for Earth – the 'D' in D-PROV), a large NSF data federation and preservation project which provides a large scale data infrastructure where member nodes host data packages. These are bundles of data artifacts of different kinds, annotated using various types of metadata, which scientists can upload, search through, and reuse. Scientists who use DataONE can use data packages as a way to combine datasets, specifications of the experimental methods used to produce them, specifically workflows, along with provenance traces obtained as a result of the workflow executions. Scientists can then discover datasets of interest by exploiting the rich and structured metadata offered by provenance traces.

To motivate the need for extended provenance traces in this setting, consider the workflow of Figure 1, implemented using the VisTrails[3] workflow system [19] for re-gridding climate datasets [18]. The workflow proceeds as follows. First, the *ParseData* module reads a benchmark data file, and parses it into a form that can be consumed by VisTrails modules. A benchmark data file contains monthly data for 12 months. The data produced by the *ParseData* module is filtered to extract North American climate data using the *Subset* module. The *Regrid* module then modifies the resolution of data from 0.5-degrees to 1-degree, and the *ConvertUnits* module converts the unit of data values from gC/m^2/day to kgC/m^2/month[4]. The obtained data is visualized using the *VisualizeData* module, and stored locally using the *SaveData2Local* module.

We are interested in answering queries on a collection of provenance traces corresponding to executions of workflows such as the one above. Some of the queries are

[1] http://www.w3.org/TR/prov-dm/
[2] http://www.dataone.org
[3] http://www.vistrails.org
[4] Carbon concentration in grams per square meter per day to carbon concentration in kilograms per square meter per month.

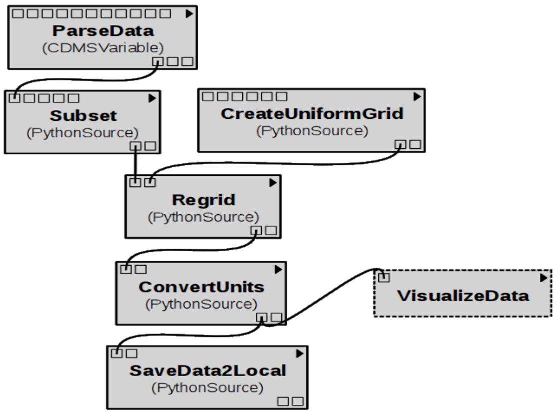

Figure 1: VisTrails workflow for *re-gridding* climate data

not concerned with workflow structure and thus they can be answered using PROV alone, for instance *"track the lineage of the final outputs of the workflow"* (Q1). Others, however, require the workflow structure, e.g., *"list the parameter values that were used for a specific task t in the workflow"* (Q2), and *"check that the provenance traces conform to the structure of the workflow"* (Q3).

D-PROV, the PROV extension sketched in the rest of the paper, provides scientists with a vocabulary and relational structure that makes it possible to answer queries like Q2 and Q3. A Datalog specification of these queries over D-PROV traces is provided in Section 4.

We distinguish between three levels of workflow provenance information, using nomenclature introduced by Feire *et al.* [5] and Lim *et al.* [11]:

1. *Retrospective Provenance (r-prov)*: refers to the provenance of the data produced by one run of a process (workflow) P ;

2. *Prospective Provenance (p-prov)*: refers to the representation of the process/workflow P itself;

3. *Provenance of the Process*: refers to an account of the evolution of P across versions.

D-PROV is designed to capture retrospective and prospective provenance information, that is (1) and (2). Part (3) is beyond the scope of this short paper. We present D-PROV in Section 3. Furthermore, we discuss several issues that arise when collecting both prospective and retrospective provenance in Section 4, namely (i) the conformance of retrospective provenance of a given workflow run with its corresponding prospective provenance (i.e., process definition), (ii) the similarity between the retrospective provenance of multiple workflow runs, and show how they can be answered using D-PROV.

2 Related work

The representation of workflow structure used in D-PROV is inspired primarily by dataflow models commonly found in scientific workflow systems such as Kepler, Taverna, VisTrails, and others [12, 15, 19, 3], and builds on the earlier experience of the Janus model of provenance [14]. The latter defines an ontology to model both retrospective and prospective provenance for the Taverna workflow system [8]. In a related earlier effort, Garijo and Gil [6] used the Open Provenance Model [16] to represent at the same time r-prov as well as p-prov statements, in the context of Linked Open Data. As this is done without introducing any extended vocabulary, however, the result is an overloading of some of the OPM terms (for instance "Process" is used both to represent execution and process templates). We note that an earlier version of D-PROV, described using UML and without making a connection with PROV, appears in [2].

The work by the Wf4Ever team[5] around research objects [1] is perhaps the closest to ours. In essence, a research object is a bundle that aggregates a number of resources that are necessary for understanding, reusing and reproducing the results of research investigations. In particular, the research object model provides two vocabularies, *wfdesc* and *wfprov*, for specifying the prospective and retrospective provenance of workflows, respectively. *wfdesc* and *wfprov* share some similarities with D-PROV, e.g., they extend the W3C PROV model. However, they are fundamentally different from D-PROV for the two following reasons: (i) D-PROV has been designed with the objective to act as a global model that caters for most of the constructs in major workflow models. Conversely, the focus in *wfdesc* and *wfprov* has been on capturing the minimal constructs that are common to data-driven workflows. For example, D-PROV supports both channel- and port-based workflows, whereas *wfdesc* and *wfprov* are confined to port-based workflows as it is the model adopted by the majority of scientific workflow systems. (ii) D-PROV targets workflows that are executable, i.e., workflows in which the steps are implemented (at least partly) by software components. *wfdesc* and *wfprov*, on the other hand, are used to model executable workflows as well as abstract workflows, which can be used, for instance, to document the method followed by scientists in their investigations.

3 PROV and D-PROV

A dataflow like the one in Figure 1 can be viewed as a directed (sometimes acyclic) graph whose nodes are tasks, i.e., units of computation, and whose arcs are interpreted

[5] http://www.wf4ever-project.org

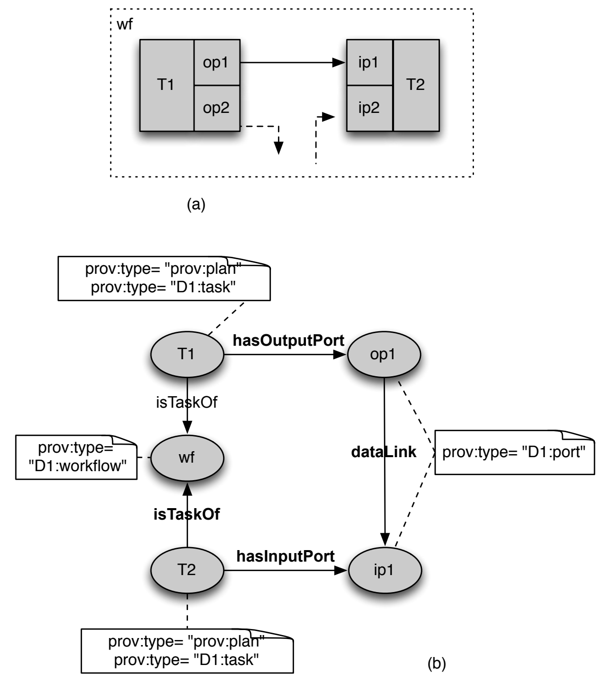

Figure 2: Port-oriented workflow pattern and corresponding D-PROV representation

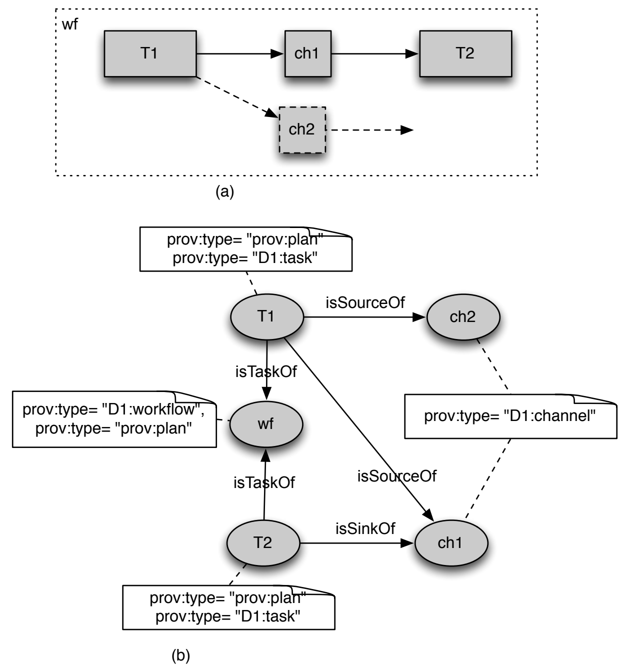

Figure 3: Channel-oriented workflow pattern and corresponding D-PROV representation

as data dependencies amongst tasks. Workflow models that follow this fundamental dataflow structure may differ in their definition of data dependencies, and on the specific dataflow semantics. In systems like Kepler, Taverna, VisTrails, and e-Science Central [7] amongst others, each task has a set of input and a set of output ports which define a task's interface. A data dependency arc in the dataflow graph connects one output port *op* of task T_1 to one input port *ip* of task T_2 (Fig. 2(a)).[6] Thus, in a programming sense tasks play the role of functions, and ports are formal input/output arguments. Tasks consume values as actual parameters on their input ports, and produce output values bound to the output ports. The arcs denote how these values flow from one task to another in the network. In these systems, provenance is typically recorded by observing the invocation of tasks, as well as the data values that appear on their ports. Tasks are modelled as PROV *activities*, while data values are *entities*.

In scientific workflow systems that are based on Kahn process networks [9, 10], notably Kepler, we can alternatively think of workflows as bipartite graphs in which tasks (called *actors*) communicate with each other through (unidirectional FIFO) *channels*. Thus, directed edges are either from actor to channel (output edge), or from channel to actor (input edge). This is illustrated in Fig. 3(a). In this model, the observable trace events include actor invocations, and data read-from or written-to a channel. In the following, we use both these model variations as reference.

Regarding r-prov, PROV offers core relations to represent that an entity (a data item) was generated by or was used by an activity (a task instance, or *invocation*). Relations are also available to express additional provenance semantics, namely: `wasInformedBy`, to denote that an activity depended on another through an implicit data production/usage relationship; `wasDerivedFrom`, to denote that an entity was derived from another through an implicit activity; `wasStartedBy` and `wasEndedBy`, to indicate that an activity was started (resp., ended) by another, possibly by means of a trigger. None of these relations, however, capture the graph structure of the dataflow itself, i.e., the schema information, as this is out of the scope of PROV. For that, the one available concept is that of a `plan`, a generic reference to any entity that was used by some agent whilst carrying out an activity. For instance, a `plan` can be a software program, a cooking recipe, on anything else that describes *how* an activity was carried out. A plan can be used as part of ternary relation:

```
wasAssociatedWith(a, ag, plan)
```

where `a` is an activity, and `ag` is an optional reference

[6] This core model may include additional elements, such as a variety of control dependencies amongst tasks (for instance, T_2 may only start after T_1 completes) and structural nesting of worfklow structure, whereby a task may itself be a workflow.

Entity types:	D1:workflow, D1:port, D1:task, D1:channel
p-prov Relations:	
taskOf(t, wf)	task t is part of workflow wf
hasOutPort(t,p)	task t has output port p
hasInPort(t,p)	task t has input port p
dataLink(p1, p2)	a data link connects port p1 to p2
sourceOf(t,c)	task t is the source of channel c
sinkOf(t,c)	task t is the sink of channel c
r-prov Relations:	
onInPort(d, p, tInv)	data d was observed on input port p
onOutPort(d, p, tInv)	data d was observed on output port p
wasWrittenTo(d,c,tInv)	data entity d was written to channel c
wasReadFrom(d,c,tInv)	data entity d was read from channel c

Table 1: D-PROV extensions for workflow-specific p-provenance and r-provenance

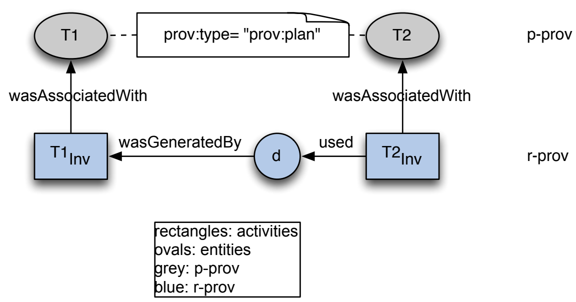

Figure 4: Minimal representation of retrospective provenance generated from program execution, expressed using the core PROV vocabulary

to an agent who has been responsible for carrying out a. PROV also provides the built-in entity type `prov:plan` to qualify entities that can play the role of plans in such relations.

Using these built-in facilities, one can only go as far as modelling tasks as plans and using the association relation to link tasks to their invocation. The dataflow fragments of Fig. 2(a) and Fig. 3(a) and their corresponding run can both be expressed as follows:

```
% p-prov: tasks, but no data or activities
entity (t1, [ prov:type = 'prov:plan'])
entity (t2, [ prov:type = 'prov:plan'])
% r-prov - task invocation and data
activity (t1inv)
activity (t2inv)
entity (d)  % data flowing between two task instances
wasGeneratedBy(d, t1inv)
used(t2inv, d)
%% connecting r-prov and p-prov
wasAssociatedWith(t1inv, _, t1)  % t1 is the plan for t1inv
wasAssociatedWith(t2inv, _, t2)  % t1 is the plan for t2inv
```

This of course assumes that data generation and usage relations correctly represent the semantics of data manipulation by task invocations. The corresponding graph is shown in Fig. 4.

D-PROV extends this baseline provenance pattern. The extensions, listed in Table 1, consist of new entity types that qualify PROV entities, as well as new relations. Note that we use the new DataONE `D1` namespace to identify the new vocabulary. These extensions accommodate both port-oriented and channel-oriented workflow models, with distinct usage patterns for each of them. The following statements capture p-prov and r-prov for port-oriented workflows.

```
% port-oriented workflow provenance
bundle wfDesc % p-prov model
 entity (t1, [ prov:type = 'D1:task', prov:type = 'prov:plan'])
 entity (t2, [ prov:type = 'D1:task', prov:type = 'prov:plan'])
 entity (op1, [ prov:type = 'D1:port',
                D1:datatype='D1:data:climate'])
 entity (ip1, [ prov:type = 'D1:port',
                D1:datatype='D1:data:climate'])
 entity (wf,  [ prov:type = 'D1:workflow',
                prov:type = 'prov:plan'])

hasOutPort(t1, op1)
hasInPort(t2, ip1)
dataLink(op1, ip1)
isTaskOf(wf, t1)
isTaskOf(wf, t2)
endbundle

bundle wfRunTrace %% r-prov model
 activity (wfRun)
 activity (t1inv)
 activity (t2inv)
wasStartedBy(t1Inv, wfRun)
wasStartedBy(t2Inv, wfRun)
 entity (d)
onOutPort(d, op1, t1Inv)
onInPort(d, ip1, t2Inv)

%% connecting r-prov and p-prov
wasAssociatedWith(wfRun, _, wf)  % wf is the plan for wfRun
wasAssociatedWith(t1inv, _, t1)  % t1 is the plan for t1inv
wasAssociatedWith(t2inv, _, t2)  % t1 is the plan for t2inv
endbundle
```

The p-prov portion of these statements is depicted in the graph of Fig. 2(b), while the p-prov and r-prov statements are combined in the graph of Fig. 5. PROV-aware systems that are not D-PROV aware can use the corresponding, less informative plain PROV statements involving generation/usage relations, obtained by means of two simple inference rules for *port elimination*, written in Datalog style:

```
wasGeneratedBy(D, tInv) :- onOutPort(D, _, tInv).
used(tInv, D) :- onInPort(D, _, tInv)
```

Regarding channel-oriented workflows, the following statements capture the corresponding provenance fragment. The p-prov portion appears in Fig. 3(b), while the combined p-prov, r-prov statements are in Fig. 6:

```
% channel-oriented workflow provenance
bundle wfDesc % p-prov model
 entity (t1, [ prov:type = 'D1:task', prov:type = 'prov:plan'])
 entity (t2, [ prov:type = 'D1:task', prov:type = 'prov:plan'])
```

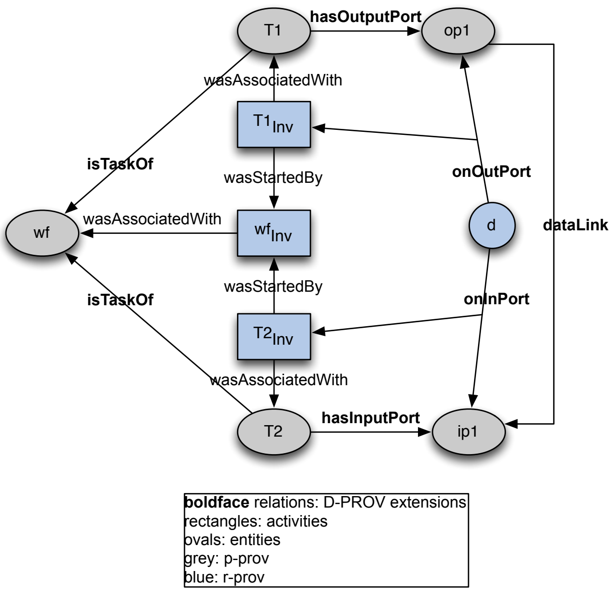

Figure 5: Graph representation of p- and r-provenance for port-oriented workflow

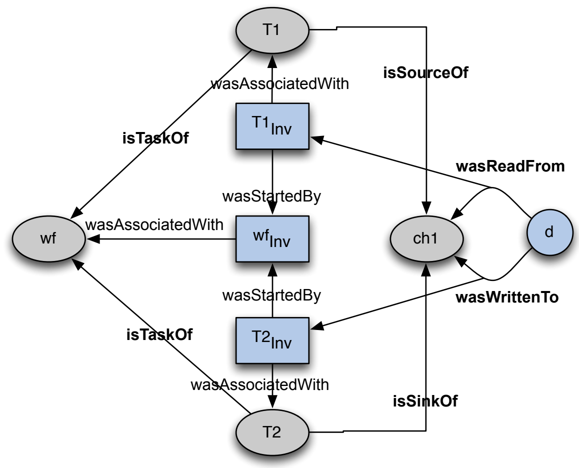

Figure 6: Graph representation of p- and r-provenance for channel-oriented workflow

```
 entity (ch,   [ prov:type = "D1:channel",
                 D1:datatype='D1:data:climate'])
 entity (wf,   [ prov:type = 'D1:workflow',
                 prov:type = 'prov:plan'])
sourceOf(t1 ,ch)
sinkOf(t2 ,ch)
isTaskOf(t1 ,  wf)
isTaskOf(t2 ,  wf)
endbundle

bundle  wfRunTrace  %% r−prov model
 activity  (wfRun)
 activity  ( t1inv )
 activity  ( t2inv )
 entity (d)
wasWrittenTo(d,ch,  t1Inv )
wasReadFrom(d,ch, t2Inv)

%% connecting r−prov and p−prov
wasAssociatedWith(wfRun, _,  wf)   % wf is  the  plan  for  wfRun
wasAssociatedWith(t1inv ,  _,  t1 )   % t1  is  the  plan  for  t1inv
wasAssociatedWith(t2inv ,  _,  t2 )   % t1  is  the  plan  for  t2inv
endbundle
```

Again, one can remove the extensions from r-prov by means of simple rules:

```
wasGeneratedBy(d, tInv )  :− wasWrittenTo(d,ch,  t1Inv ).
used( tInv ,  D) :− wasReadFrom(d,ch, t2Inv)
```

Bundles and the provenance traces of sub-workflows. In the listings above, note the use of *bundles* as a way to group provenance statements. Bundles are introduced in the PROV specifications to support expressing *provenance of provenance*, that is, for expressing the provenance of a set of provenance statements. In this setting, we use them instead to group together provenance statements that pertain to p-prov, or to the r-prov for a single workflow run, as in bundle `wfRunTrace` above. Furthermore, in PROV a named bundle of provenance statements is itself an entity, of type `prov:bundle`. This makes it possible to establish an explicit association between a workflow execution and the provenance it generates, by stating that a workflow run generated an entire provenance trace, as follows:

```
 entity (wfRunTrace, [  prov:type='prov:Bundle' ])
wasGeneratedBy(wfRunTrace, wfRun, _)
```

We leverage this bundle naming and referencing facility to model the provenance of hierarchically structured workflows, in a recursive fashion. Suppose for instance that task `t2` is itself a workflow, i.e., it is of type `D1:workflow` as well as `D1:task`:

```
 entity (t2 ,  [  prov:type  =  "D1:task", prov:type  =  "D1:workflow" ])
```

Then, one execution of `t2inv`, an invocation of `t2`, may generate a provenance trace within a bundle, say `t2invTrace`. Such a bundle may now be referenced as part of a larger bundle that represents the entire workflow execution, i.e.:

```
bundle  wfRunTrace
 activity  (wfRun)
 activity  ( t1inv )
 activity  ( t2inv )
 ...
 entity ( t2invTrace ,  [  prov:type='prov:Bundle' ])
wasGeneratedBy(t2invTrace,  t2inv ,  _)
 ...
endbundle
```

This shows how one can at the same time represent structural containment within a workflow, i.e., by using `prov:type = "D1:task"`, `prov:type = "D1:workflow"`, and associate whole traces to each subworkflow.

4 Query Answering

Having introduced the necessary relations, we now describe the answers of the queries presented in Section 1. Following [4], we use Datalog to express provenance queries (the PROV-N syntax used in this paper has been shown [13] to map easily to Datalog).

Query 1: The *ConvertUnits* process generated the *climate data in kgC/m²/month*, resulting in the following output:

```
entity (op, [prov:typ='D1:port',
        D1:datatype='D1:data:climate:kgC/m^2/month']).
```

As this entity relation has an optional list of attributes, we normalize this entity relation using Datalog as follows, as shown in [13]:

```
entity (EntityID, AttrListID ).
attr ( AttrListID , AttrName, AttrValue ).
```

In a port-oriented workflow, the lineage of this final output can be obtained using the following query, which predicates on the values of the attributes associated to the entity of interest:

```
dep(D,I) :− onInPort(D,_,I ).
dep(I,D) :− onOutPort(D,_,I ).
lineage (X,Y) :−
  dep(X,Y), onOutPort(Y,P,_),  entity (P,A),
  attr (A,'D1:datatype','D1:data:climate:kgC/m^2/month').
lineage (X,Y) :−
  dep(X,Y), lineage (Y,Z).
```

The `lineage` relation produces a subgraph in which the final data product is dependent on all nodes except the leaves.

Query 2: The set of parameter values that were used for a specific task in the workflow, is defined by the following Datalog program:

```
parameter(T,D) :−
  onInPort(D,P,I ),  entity (P,A),
  attr (A,'D1:input:type','parameter'),
  wasAssociatedWith(I,_,T).
```

Here, the `parameter` relation associates data identifiers to tasks. Data values are obtained simply by dereferencing these identifiers.

Query 3 is about verifying that the structure of a trace conforms to the structure of its specification. This involves two steps: firstly, to add rules that entail p-prov relations from r-prov relations, and secondly, to check that those new p-prov relations are consistent with any constraints defined on the workflow structure. The following is an example of the first step:

```
t_dataLink (OP, IP) :− onOutPort(D,OP,I1),  onInPort(D,IP,I2 ),
   wasAssociatedWith(I1,_,T1),  wasAssociatedWith(I2,_,T2),
   isTaskOf(T1, Wf), isTaskOf(T2, Wf).
```

The rule states that if the same data item D is observed on an output and on an input port in a context where the corresponding tasks are part of the same workflow, then there must be a data link between the two ports. Using an open world assumption, a p-prov specification in which such data link is missing can be considered incomplete, and thus the new relation can be added to the specification, whilst in a closed world, one would conclude that the trace does not conform to the specification.

Similarly, one may entail a dependency amongst tasks, from r-prov observations rule (1), below, and check those against statically determined dependencies (rule (2)):

```
% (1)
t_taskDep(T2,T1) :− onOutPort(D,P1,I1 ),  onInPort(D,P2,I2 ),
  wasAssociatedWith(I1,_,T1),  wasAssociatedWith(I2,_,T2).
% (2)
s_taskDep(T2,T1) :− hasOutPort(T1,P1),  hasInPort (T2,P2),
                    dataLink(P1,P2).
```

5 Conclusions

We have presented D-PROV, an extension to the W3C PROV specification, which accounts for structural features of typical dataflow models, namely those where inter-task communication is either port-based or channel-based. Following [5], we have referred to them as *prospective provenance*. D-PROV has been defined in the context of the DataONE Provenance Working Group. The intent of D-PROV is to enable queries that blend together prospective and retrospective provenance. We have motivated such need and shown some of the benefits of D-PROV, by means of example queries, which we situated in the context of scientific workflows defined by DataONE scientists.

Acknowledgments. Work supported in part by NSF-OCI DataONE #0830944 (for Víctor Cuevas-Vicenttín) and made possible by the voluntary work of members of the DataONE Provenance Working Group. Special thanks to Yaxing Wei from ORNL for the design and implementation of the VisTrails climate workflows. Khalid Belhajjame was supported by the myGrid platform grant.

References

[1] BELHAJJAME, K., CORCHO, O., GARIJO, D., ZHAO, J., MISSIER, P., NEWMAN, D., PALMA, R., BECHHOFER, S., GARCÍA CUESTA, E., GÓMEZ-PÉREZ, J. M., KLYNE, G., PAGE, K., ROOS, M., RUIZ, J. E., SOILAND-REYES, S.,

VERDES-MONTENEGRO, L., DE ROURE, D., AND GOBLE, C. A. Workflow-Centric Research Objects: First Class Citizens in Scholarly Discourse. In *SePublica2012 workshop at ESWC2012* (2012).

[2] CUEVAS, V., DEY, S., AND LUDÄSCHER, B. Modeling and Querying Scientific Workflow Provenance in the D-OPM. In *Workshop on Workflows in Support of Large-Scale Science (WORKS)* (2012).

[3] DEELMAN, E., GANNON, D., SHIELDS, M., AND TAYLOR, I. Workflows and e-Science: An overview of workflow system features and capabilities. *Future Generation Computer Systems 25*, 5 (2009), 528–540.

[4] DEY, S., KÖHLER, S., BOWERS, S., AND LUDÄSCHER, B. Datalog as a lingua franca for provenance querying and reasoning. In *Workshop on the Theory and Practice of Provenance (TaPP)* (2012).

[5] FREIRE, J., KOOP, D., SANTOS, E., AND SILVA, C. T. Provenance for Computational Tasks: A Survey. *Computing in Science and Engineering 10*, 3 (2008), 11–21.

[6] GARIJO, D., AND GIL, Y. A New Approach for Publishing Workflows: Abstractions, Standards, and Linked Data. In *Proceedings of the Sixth Workshop on Workflows in Support of Large-Scale Science (WORKS'11), held in conjunction with SC 2011* (Seattle, Washington, 2011).

[7] HIDEN, H., WOODMAN, S., WATSON, P., AND CALA, J. Developing Cloud Applications using the e-Science Central Platform. *Proceedings of Royal Society in press* (2012).

[8] HULL WOLSTENCROFT, K., STEVENS, R., GOBLE, C., POCOCK, M.R., LI, P., AND OINN, T., D. Taverna: a tool for building and running workflows of services. *Nucleic Acids Research*, 34(Web Server issue) (2006), W729—-W732.

[9] KAHN, G. The semantics of a simple language for parallel programming. In *Information Processing* (Stockholm, Sweden, Aug 1974), J. L. Rosenfeld, Ed., North Holland, Amsterdam, pp. 471–475.

[10] LEE, E. A., AND PARKS, T. M. Dataflow process networks. *Proceedings of the IEEE 83*, 5 (1995), 773–801.

[11] LIM, C., LU, S., CHEBOTKO, A., AND FOTOUHI, F. Prospective and Retrospective Provenance Collection in Scientific Workflow Environments. In *Services Computing (SCC), 2010 IEEE International Conference on* (July 2010), pp. 449–456.

[12] LUDÄSCHER, B., ALTINTAS, I., BERKLEY, C., HIGGINS, D., JAEGER, E., JONES, M., LEE, E. A., TAO, J., AND ZHAO, Y. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience 18*, 10 (2006), 1039–1065.

[13] MISSIER, P., AND BELHAJJAME, K. A PROV encoding for provenance analysis using deductive rules. In *Procs. IPAW'12* (Santa Barbara, California, 2012), Springer-Verlag, Lecture Notes in Computer Science.

[14] MISSIER, P., SAHOO, S. S., ZHAO, J., SHETH, A., AND GOBLE, C. Janus: from Workflows to Semantic Provenance and Linked Open Data. In *Procs. IPAW 2010* (Troy, NY, 2010).

[15] MISSIER, P., SOILAND-REYES, S., OWEN, S., TAN, W., NENADIC, A., DUNLOP, I., WILLIAMS, A., OINN, T., AND GOBLE, C. Taverna, reloaded. In *Procs. SSDBM 2010* (Heidelberg, Germany, 2010), M. Gertz, T. Hey, and B. Ludäscher, Eds.

[16] MOREAU, L., CLIFFORD, B., FREIRE, J., FUTRELLE, J., GIL, Y., GROTH, P., KWASNIKOWSKA, N., MILES, S., MISSIER, P., MYERS, J., PLALE, B., SIMMHAN, Y., STEPHAN, E., AND VAN DEN BUSSCHE, J. The Open Provenance Model — Core Specification (v1.1). *Future Generation Computer Systems 7*, 21 (2011), 743–756.

[17] 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. Tech. rep., World Wide Web Consortium, 2012.

[18] SANTOS, E., POCO, J., WEI, Y., LIU, S., COOK, R., WILLIAMS, D., AND SILVA, C. T. UV-CDAT: Analyzing Climate Data sets from a Users Perspective. *Computing in Science and Engineering 15* (2013), 94–103.

[19] SCHEIDEGGER, C. E., VO, H. T., KOOP, D., FREIRE, J., AND SILVA, C. T. Querying and re-using workflows with vistrails. In *SIGMOD* (2008), ACM, pp. 1251–1254.