D-PROV: Extending the PROV Provenance Model to express Workflow Structure

Paolo Missier\(^{(1)}\), Saumen Dey\(^{(2)}\), Khalid Belhajjame\(^{(3)}\), Víctor Cuevas-Vicenttí\(^{(2)}\), Bertram Ludaescher\(^{(2)}\)

\(^{(1)}\) Newcastle University, UK
\(^{(2)}\) UC Davis, CA, USA
\(^{(3)}\) University of Manchester, UK

TAPP’13
Lombard, IL.
April, 2013
Queries that require provenance

Q1: “track the lineage of the final outputs of the workflow”
Q2: “list the parameter values that were used for a specific task t in the workflow”
Q3: “check that the provenance traces conform to the structure of the workflow”

Prospective Provenance (p-prov):
- representation of the workflow itself;

Retrospective Provenance (r-prov):
- provenance of the data produced by one workflow run

Provenance of the Process:
- account of the evolution of the workflow across versions

PROV @W3C: scope and structure

Recommendation track

- PROV-O
- PROV-XML
- Other Serialization
- PROV-DC
- PROV-LINKS
- PROV-AQ
- PROV-CONSTRAINTS
- PROV-DM
- PROV-N

source: http://www.w3.org/TR/prov-overview/
r-prov and p-prov in plain PROV

// 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
Reference dataflow models

Processors, ports, data links

Kepler
Taverna
VisTrails
e-Science Central
...

Processors, channels

Dataflow process networks (*)
(e.g. a specific Kepler semantics)

Extensions I / p-prov / ports model

```
prov:type= "prov:plan"
prov:type= "D1:task"

prov:type= "D1:workflow"

hasOutPort(T1, op1)
hasInPort(T2, ip1)
dataLink(op1, ip1)
isTaskOf(wf, T1)
isTaskOf(wf, T2)
```
Extensions II / p-prov / channel model

prov:type= "prov:plan"
prov:type= "prov:plan"
prov:type= "D1:task"
prov:type= "D1:channel"

isTaskOf(T1, wf)
isTaskOf(T2, wf)
isSourceOf(T1, ch2)
isSourceOf(T1, ch1)
isSourceOf(T2, ch1)
p-prov/r-prov pattern for port-oriented workflows

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

activity (wfRun)
activity (t1inv)
activity (t2inv)
entity (d)
onOutPort(d, op1, t1Inv)
onInPort(d, ip1, t2Inv)
Lossy mapping to plain PROV: Port removal

\[
\text{wasGeneratedBy}(D, \text{tInv}) \leftarrow \text{onOutPort}(D, \_, \text{tInv}) .
\]

\[
\text{used} (\text{tInv}, D) \leftarrow \text{onInPort}(D, \_, \text{tInv})
\]
The image shows a diagram illustrating the p-prov/r-prov pattern for channel-oriented workflows. The diagram contains nodes and labeled connections indicating different types of relationships and activities. Here is the textual representation of the diagram:

- **Source Of** ($\text{sourceOf}(t1, ch)$)
- **Sink Of** ($\text{sinkOf}(t2, ch)$)
- **Is Task Of** ($\text{isTaskOf}(t1, \text{wf})$)
- **Is Task Of** ($\text{isTaskOf}(t2, \text{wf})$)
- **Activity** ($\text{activity}(\text{wfRun})$)
- **Activity** ($\text{activity}(t1\_\text{inv})$)
- **Activity** ($\text{activity}(t2\_\text{inv})$)
- **Entity** ($\text{entity}(d)$)
- **Was Written To** ($\text{wasWrittenTo}(d, ch, t1\_\text{inv})$)
- **Was Read From** ($\text{wasReadFrom}(d, ch, t2\_\text{inv})$)

The diagram includes nodes labeled with task names and workflow, channel, and entity concepts, connected with arrows indicating the flow of information and relationships between these concepts.
p-prov/r-prov pattern for channel-oriented workflows

Lossy mapping to plain PROV: Channel removal:

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

sourceOf(t1, ch)
sinkOf(t2, ch)
isTaskOf(t1, wf)
isTaskOf(t2, wf)
activity (wfRun)
activity (t1inv)
activity (t2inv)
entity (d)
wasWrittenTo(d, ch, t1Inv)
washReadFrom(d, ch, t2Inv)
A bundle is a named set of provenance descriptions, and is itself an entity, so allowing provenance of provenance to be expressed.

```
bundle pm:bundle1

entity(ex:draftComments)
entity(ex:draftV1)

activity(ex:commenting)
wasGeneratedBy(ex:draftComments, ex:commenting,-)
used(ex:commenting, ex:draftV1, -)
endBundle
...
entity(pm:bundle1, [ prov:type='prov:Bundle' ])
wasGeneratedBy(pm:bundle1, -, 2013-03-20T10:30:00)
wasAttributedTo(pm:bundle1, ex:Bob)
```
Repurposing: use bundles to associate a workflow execution with the provenance it generates

```
entity (wfRunTrace, [ prov:type='prov:Bundle' ])  
wasGeneratedBy(wfRunTrace,wfInv,-)
```

This makes it possible to write hierarchical provenance of nested workflows, recursively:

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

```
bundle wfRunTrace  
  activity(wfRun)  // run of top level wf  
  activity(Tinv)  // run of T, a sub-workflow
  wasAssociatedWith(Tinv, _, T)  
  entity(TinvTrace, [ prov:type='prov:Bundle' ]) 
  wasGeneratedBy(TinvTrace, Tinv, _) 
  ...
endbundle
```
Q3: “match the provenance trace to the workflow structure”

Two steps:
- define rules that entail p-prov relations from r-prov relations, and
- check that those new p-prov relations are consistent with any constraints defined on the workflow structure / infer new p-prov statements

\[
dataLink (OP, IP) :- \text{onOutPort}(D,OP,\_), \\
\text{onInPort}(D,IP,\_).
\]
hasOutPort(T, OP) :-
  outPort(D,OP,I1),
  wasAssociatedWith(I1,_,T1).

hasInPort(T, IP) :-
  inPort(D,IP,I1),
  wasAssociatedWith(I1,_,T1).
isTaskOf(T2, Wf) :-
  onOutPort(D, OP, I1),
  hasOutPort(T1, OP),
  onInPort(D, IP, I2),
  hasInPort(T2, IP),
  wasAssociatedWith(I1, _, T1),
  wasAssociatedWith(I2, _, T2),
  isTaskOf(T1, Wf).
Prov-Wf:


Other PROV extensions into “workflow-land”

**Figure 1** PROV-Wf data model

WfProv from the Wf4Ever project  
www wf4ever-project org
Summary and extensions

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

- Simple extensions to PROV
  - designed to model p-prov
  - complementary to r-prov

- They enable queries that cut across r-prov and p-prov

- Bundle mechanism used for provenance of nested workflow components

- Next step: harmonize similar extensions proposed by other groups
  - overall goal is to achieve interoperability