Automatic vs Manual Provenance Abstractions: Mind the Gap

Pınar Alper
TAPP
9 June 2016

Pınar ALPER
University of Manchester

Khalid BELHAJJAME
Université Paris Dauphine

Carole A. GOBLE
University of Manchester
Transparency brought by Provenance can be a double edged sword

Too revealing

Secured Provenance
Transparency brought by Provenance can be a double edged sword

Too complex

Simplified Provenance
Our Goal & Context

Compare **Manual** and **Semi-automated** abstractions that **simplify workflows** in the context of reporting data-oriented experiments.

– Reports are experimental metadata on:
  - Method
  - Data

Workflow complexity necessitates abstraction

- Up to 50+ data processing tasks.
- Majority (70%) dedicated to data adaptation
- Leads to complex provenance.

Current Approach

- Manual, Design Abstractions observable in existing workflows
- Embedded into design, static

Sub-workflows

- Workflow input ports: pdfDirectoryPathIn, cValueThreshold
- Steps: RetrieveContent, ExtractText, CleanText, GetSentences, TermExtraction
- Workflow output ports: termCandidatesAboveThreshold, allTermCandidates

Diagram showing 5 Steps and 19 Steps.
Design Abstractions

Bookmarked Intermediaries
Alternative Approach

Semi-automated abstraction systems:
– dynamic
– Workflow Summaries & ZOOM UserViews

Workflow Abstraction Primitives

**Task grouping**

Task A
- inA
- outA
- inB
- outB

Task B
- inB
- outB

Task A & B
- inA
- outB

**Task elimination**

Task A
- inA
- outA
- inB
- outB
- inC
- outC

Task B
- inB
- outB

Task C
- inC
- outC

Task A
- inA
- outA

indirect df
Integrity Policy: Soundness

In the context of reporting, soundness can be compromised.

Sub-workflow based design abstractions do not necessarily preserve soundness.
Integrity Policy: Acyclicity

From a modelling perspective cycles are allowed in provenance.
Cyclic dataflows unfold into Acyclic Lineage

Raw workflow provenance is acyclic.

In the context of reporting (e.g. data tables, design abstractions) cycles are not observed.
Integrity Policy: Bipartiteness

Workflow provenance is bi-partite.

Design abstractions preserve bipartiteness.

In the context of reporting bi-partiteness is advised but not always achieved.
Design abstractions preserve validity.

In the context of reporting validity is a necessary property.
Integrity Policy: Completeness

Determined by preservation of lineage relations

Design abstractions preserve completeness

In the context of reporting completeness is a necessary property.
# Comparison of Systems

<table>
<thead>
<tr>
<th></th>
<th>Workflow Summaries</th>
<th>ZOOM</th>
<th>Design Abstractions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abstraction Policy</strong></td>
<td>Annotation-Primitive pairs</td>
<td>Important Task Ids</td>
<td>--</td>
</tr>
<tr>
<td><strong>Primitive</strong></td>
<td>Grouping</td>
<td>Elimination</td>
<td>Grouping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Grouping</td>
</tr>
<tr>
<td><strong>Integrity Policy</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Validity</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Soundness</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Bipartiteness</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Completeness</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Acyclicity</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Comparison Against Design Abstraction

- Design Abstractions as groundtruth
  - Ports on main data derivation path
  - Sub-workflow tasks.
- Workflow Summaries
  - Eliminate All Adapters
  - Collapse All Adapters
  - Collapse
- ZOOM
  - Non-Adapter tasks designated as significant
Task Elimination

<table>
<thead>
<tr>
<th>Wf Summaries -Eliminate All</th>
<th>Process Precision</th>
<th>Data Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5/5</td>
<td>0/9</td>
</tr>
</tbody>
</table>

- # of processes in the abstracted account overlapping w user’s abstraction
- total # of processes in the abstracted account

Hopping over traces does not simplify the account data –wise as as it does process-wise
A significant task and its report worthy output are not necessarily co-located
Task Grouping

ZOOM’s soundness policy creates two extra groups

Where you put the boundary to groups matters for data abstraction!
Task Grouping

Abstracting selectively (less aggressively) by taking activity function (hence I/O characteristics) into account.
Conclusions

- Abstraction systems focus on the process and do not directly cater for data significance.
- Are these observations generalisable?
  - Elimination unsuited for WFs with pack/unpack steps
  - Sweeping style grouping isn’t helpful for data abstraction
  - Selective grouping relies on domain-specific policies.
- End-use informs suitable integrity policies.
- Scientists are accountable for reports. They are likely to favor having final say on the abstraction.
- Rethink abstraction as a prehoc process supporting workflow design.

Thank You!