Data lineage model for Taverna workflows with lightweight annotation requirements

Paolo Missier, Khalid Belhajjame, Jun Zhao, Carole Goble

School of Computer Science
The University of Manchester, UK
Ongoing work on a new provenance component for Taverna
• myGrid consortium

Scope:
• capture raw provenance events
  – data transformations, data transfers
• store one *lineage graph* for each dataflow execution
• query over single or multiple *lineage graphs*
Example (Taverna) dataflow

QTL -> genes -> Kegg pathways
Some user questions on lineage

• on a single workflow run:
  – find all genes that participate in some pathway p
  – find all pathways derived from Uniprot genes
  – describe the complete derivation of each pathway in which gene g is involved

• on a collection of runs:
  – find all distinct pathways produced by runs of a dataflow
    [over a period of time, produced by a member of my group, ...]
Shortcomings of lineage data

- **Granularity**
  - risk of returning trivial answers
  - “all outputs depend on all inputs”

- **Semantics**
  - Results not expressed in the language of the designer

- **Abstraction level, noise – the “latent data model”**
  - many processors are irrelevant – shims, mundane tasks
The need for selective annotations

- As long as processors are black boxes, these remain difficult problems
- Adding annotations to processors is tempting

Scope of this work:
- to explore the “gray box” region
- simple annotations with minimal semantics
- driving principle: justified by technical benefits
  - precision of query results
  - efficiency of query processing
Test dataflow model

configuration → P₁ V₁₁ → P₁ V₁₂ → documents

P₁ extract query terms

P₂ V₁₁ → P₂ V₀₁

P₂ query prep

P₃ V₁₁ → P₃ V₀₁

P₃ query1

P₄ V₁₁ → P₄ V₀₁

P₄ query 2

P₅ V₁₁ → P₅ V₀₁

P₅ post-proc

P₆ V₁₁ → P₆ V₁₂

P₆ V₀₁ → P₆ V₀₂

P₆ merge results

number of duplicates

P₇ V₁ → P₇ V₀

P₇ sort

P₇ V₀ → merged results
Two main annotation types

**Focusing:** processor selection
- some processors are more interesting than others
  - “boring” annotations
  - query-time user selection of interesting processors

**Precision:** fine-grained lineage tracing
- goal: trace lineage of individual items within a collection
Abstraction by modularization

Lucene_query

NERecknize

extract diseases from OMIM

shims
Abstraction by selection

Workflow Inputs:
- qtl_end_position
- qtl_start_position
- chromosome_name
- genes_in_qtl

mmusculus_gene_ensembl

Operations:
- remove_uniprot_duplicates
- remove_entrez_duplicates
- create_report
- merge_uniprot_ids
- merge_entrez_genes
- merge_reports
- remove_nulls
- add_uniprot_to_string
- add_entrez_to_string
- Kegg_gene_ids
- Kegg_gene_ids_2
- concat_kegg_genes
- regex_2
- split_gene_ids
- merge_kegg_references
- split_for_duplicates
- remove_duplicate_kegg_genes

select
Abstraction by selection

Workflow Inputs:
- regex
- gene_ids

Workflow Outputs:
- pathway_genes
- pathway_desc
- pathway_ids

Other Outputs:
- merge_genes_and_pathways
- remove_pathway_duplicates
- merge_pathway_list_1
- gene_descriptions
- merge_genes_and_pathways_2
- merge_pathway_desc
- merge_pathway_list_2
- merge_gene_desc
- merge_genes_and_pathways_3
- remove_pathway_nulls
- remove_pathway_nulls_2
- remove_nulls_3

select
Focusing – processor selection

- **P1**: extract query terms
- **P2**: query prep
- **P3**: query 1
- **P4**: query 2
- **P5**: post-process
- **P6**: merge results
- **P7**: sort

**Variable Values**

<table>
<thead>
<tr>
<th>var</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P4/V1</td>
<td>a1</td>
</tr>
<tr>
<td>P1/V2</td>
<td>a2</td>
</tr>
<tr>
<td>P1/VO1</td>
<td>b</td>
</tr>
<tr>
<td>P2/V1</td>
<td>b</td>
</tr>
<tr>
<td>P4/V2</td>
<td>c1</td>
</tr>
<tr>
<td>P2/VO1</td>
<td>c2</td>
</tr>
<tr>
<td>Q/V</td>
<td>g</td>
</tr>
</tbody>
</table>

**Query**: \( \text{lineage}(P_7\ V_0, \{P_4\}) \)

**Goal**

- Avoid recursive queries on instance tables

**Idea**

- Use recursion on static model to generate a targeted query
- Execute query only once

**Note:** Assume all values atomic.
Problem: xform() also applies to list values
• It may be impossible to trace individual elements
  – “which pathways (out) depend on which genes (in)”?

Goal: extend the query generation idea just sketched to trace element-level lineage within collections

Approach: exploit static typing of Taverna processors

Taverna resolves mismatches on nesting levels:
(map P_2 [a,b,c])
Loss of precision in transformations

possible behaviours:

• selection of an element

• aggregation

function f() useful annotation:

only useful annotation:

P is index-preserving:

lineage(PV_O[i]) = f(lineage(PV_I))

PV_O[i] = PV_I[i]

lineage(PV_O[i]) = PV_I[i]
Cooperative processors

- **Passive** processors do not contribute explicit provenance info
- **Cooperative** processors actively feed metadata to the lineage service

<table>
<thead>
<tr>
<th>Static annotations:</th>
<th>Dynamic annotations:</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggregation ( f() )</td>
<td>selection: ( x = PV_i[i] )</td>
</tr>
<tr>
<td>( PV_i: I(s) = [a, b, c] )</td>
<td>( PV_O: I(s) = [x, y] )</td>
</tr>
<tr>
<td>( PV_O: s = x )</td>
<td>( PV_O[i] = PV_i[i] )</td>
</tr>
<tr>
<td>( P )</td>
<td>( \Pi(PV_i) )</td>
</tr>
</tbody>
</table>

\[ PV_O: I(s) = [x, y] \]

IPAW'08 – Salt Lake City, Utah, June 2008
Other annotations

- Distinction between configuration and input data
  - $PVI_3$ is a configuration parameter
  - compare effect of different config. across multiple runs

- specific functional dependencies
  $[ PV_{I1}, PV_{I2} ] \rightarrow PV_O$

- stateless processor
  - execute process $\leftrightarrow$ retrieve provenance

More evaluation needed on these
Towards a 2 tier provenance model

“describe the derivation of each pathway through Kegg, in which gene g is involved”

reference ontologies

semantic resource annotations

structural annotations

Semantic overlays

Lineage service

query

dataflow topology + raw lineage events

lineage database (RDB)

“describe the derivation of each pathway through Kegg, in which gene g is involved”

reference ontologies

semantic resource annotations

structural annotations

Semantic overlays

Lineage service

query

dataflow topology + raw lineage events

lineage database (RDB)

“describe the derivation of each pathway through Kegg, in which gene g is involved”

reference ontologies

semantic resource annotations

structural annotations

Semantic overlays

Lineage service

query

dataflow topology + raw lineage events

lineage database (RDB)

“describe the derivation of each pathway through Kegg, in which gene g is involved”
Conclusions

A data lineage model for Taverna workflows

- Raw lineage data has shortcomings
- A few, selected lightweight annotations added in a principled way
  - win-win:
  - helpful to users
  - *and* enable query optimization

- Form the base layer in a broader approach to efficient querying of semantic provenance for e-science
- Ongoing implementation