MANCHESTER 1824

The University of Manchester

# 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***

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

# Context and scope

Ongoing work on a new provenance component for Taverna

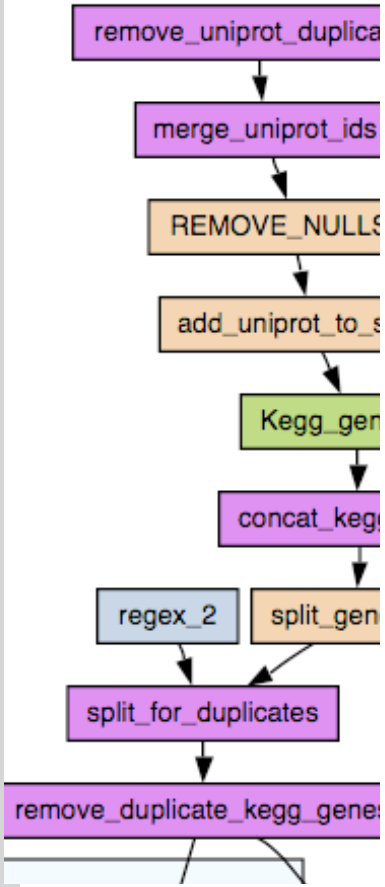
- myGrid consortium

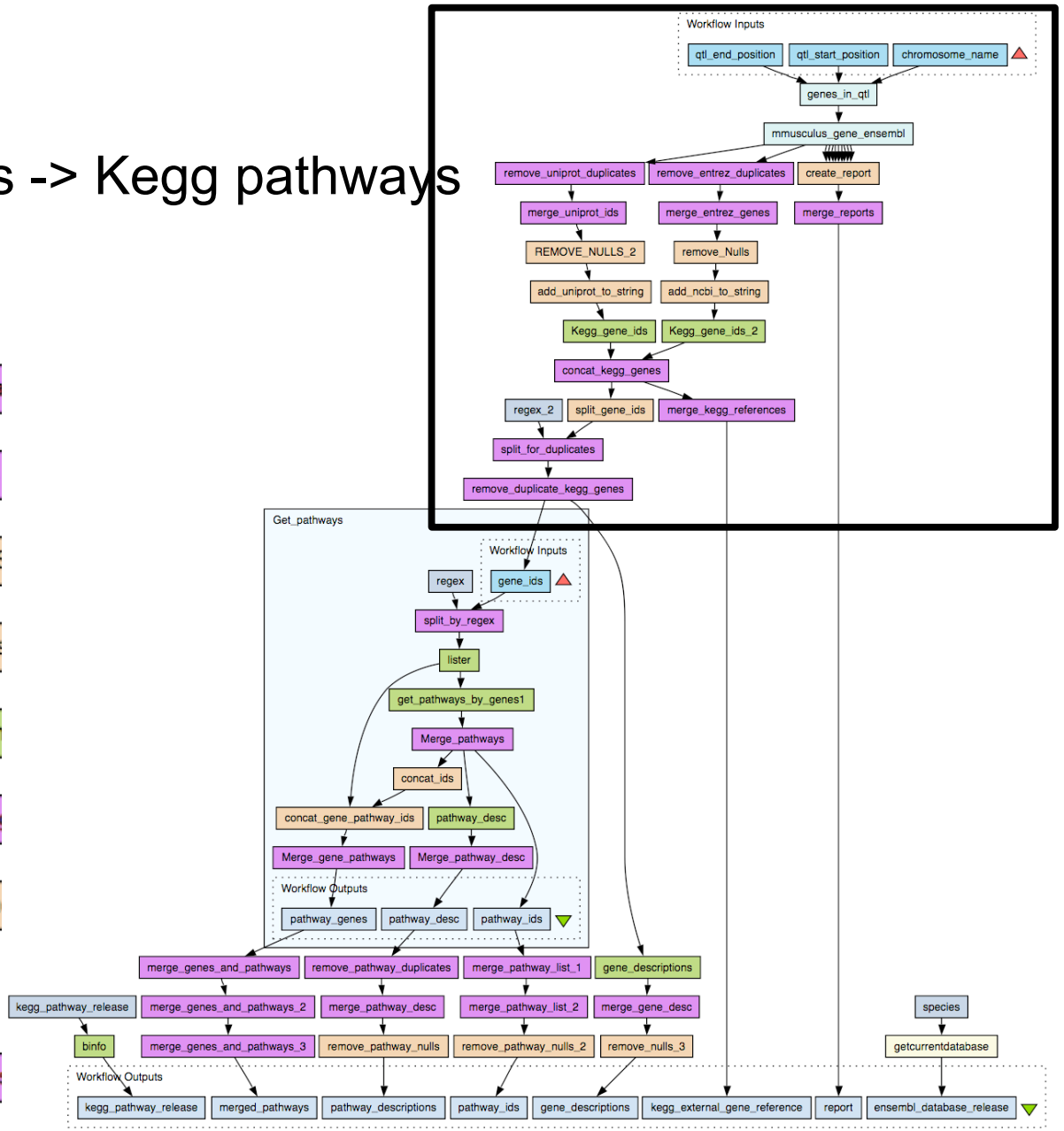
Scope:

- 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

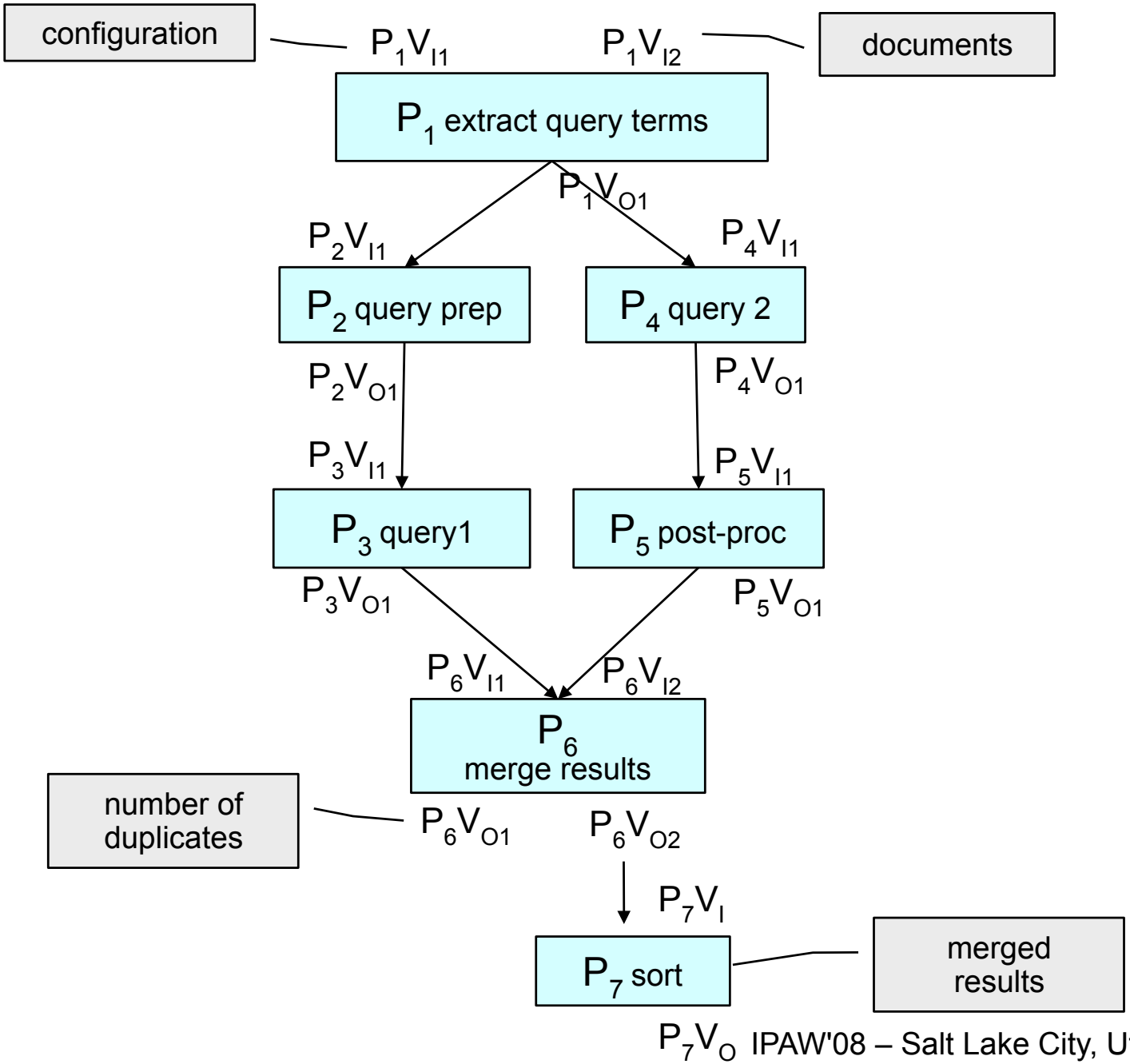
# The need for selective annotations

- 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



# Two main annotation types

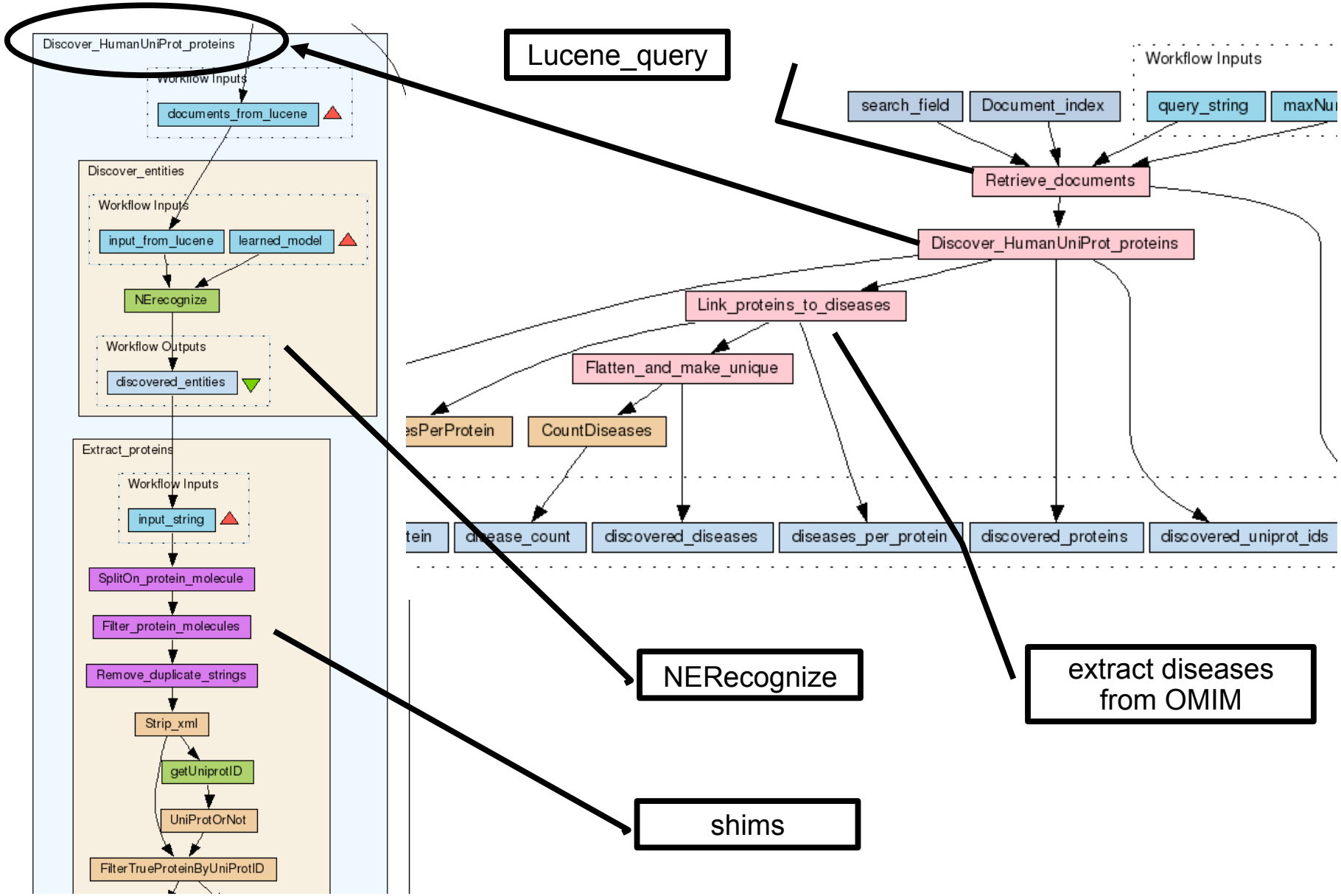
Focusing: processor selection

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

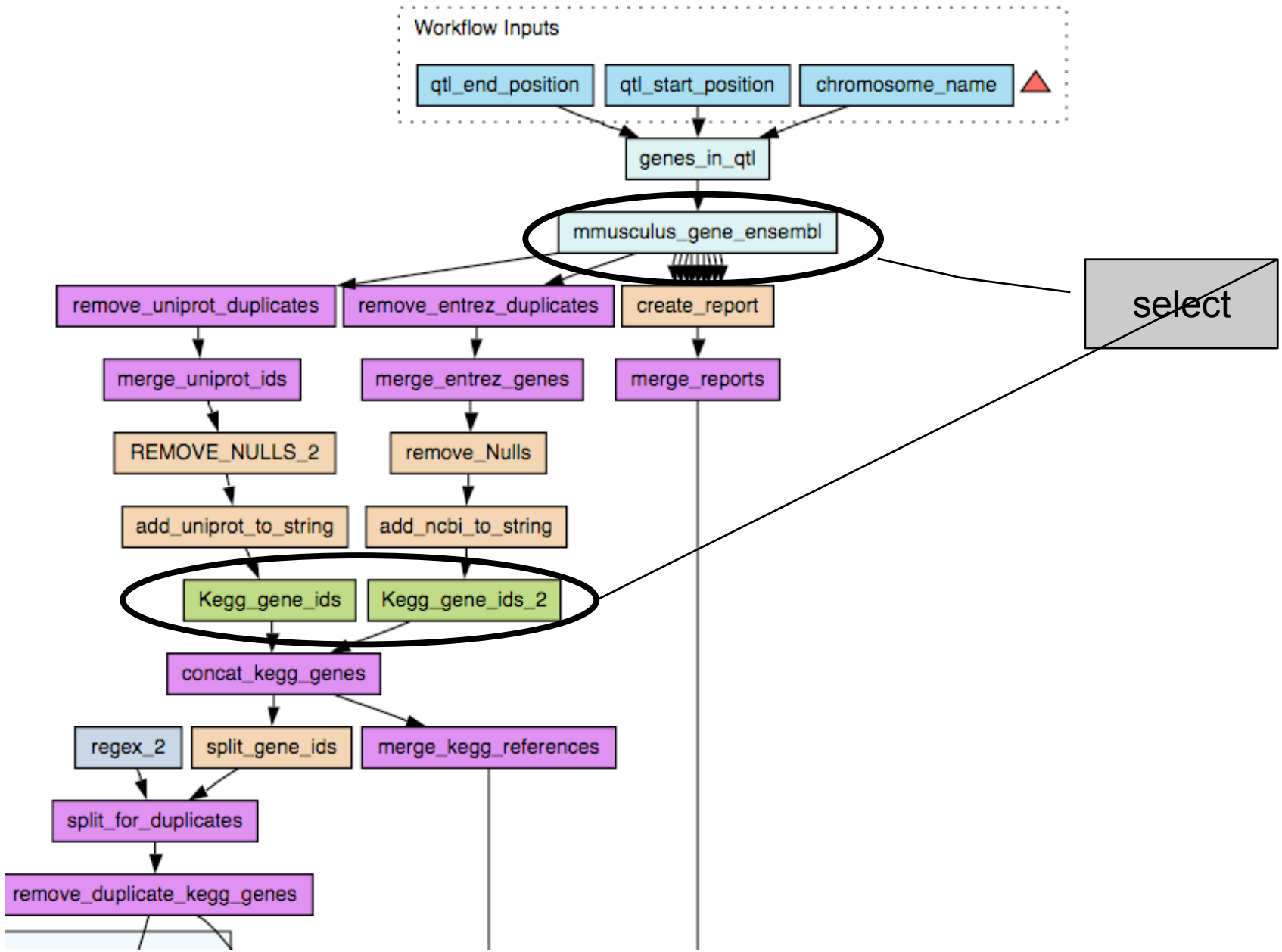
Precision: fine-grained lineage tracing

- goal: trace lineage of individual items within a collection

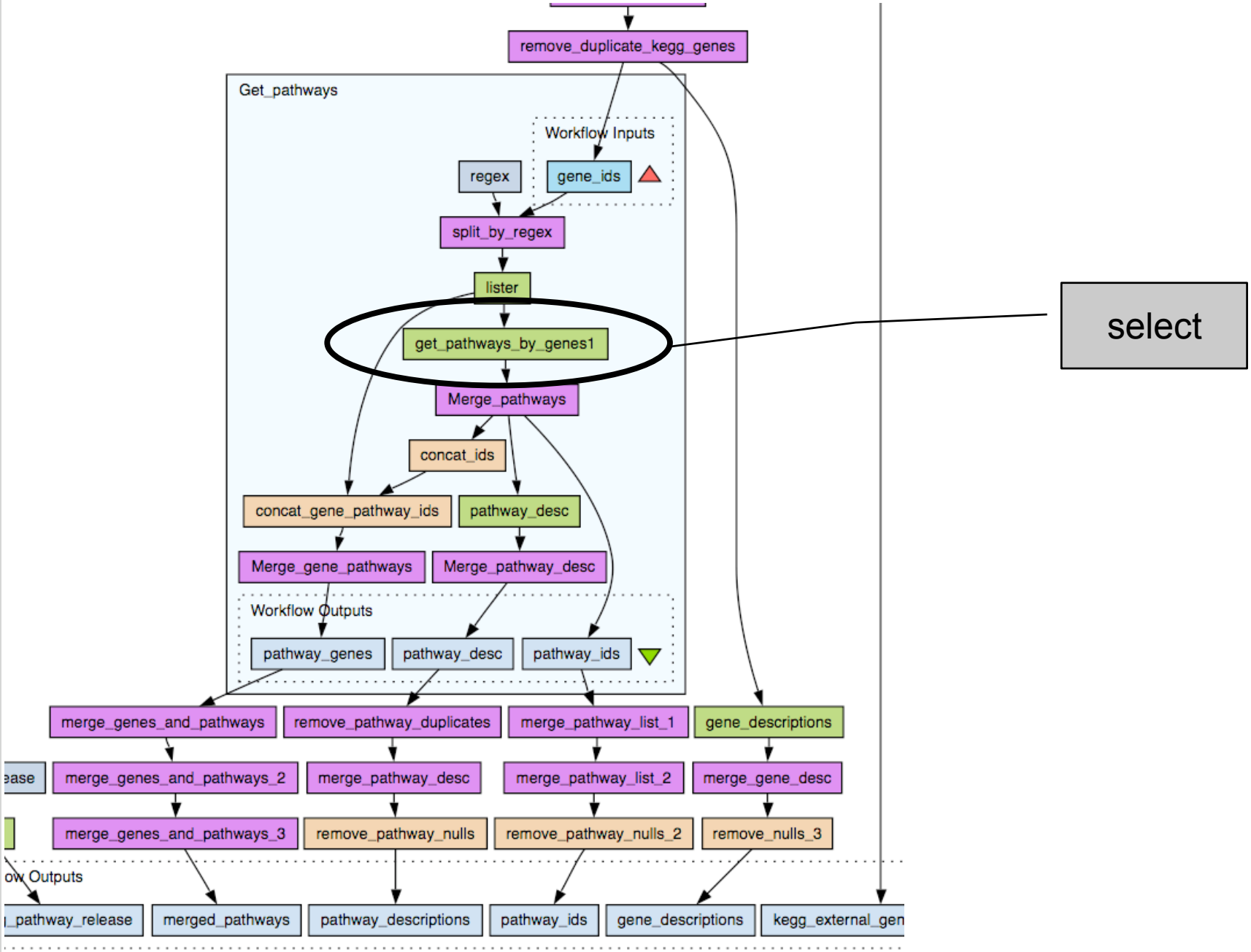
# Abstraction by modularization



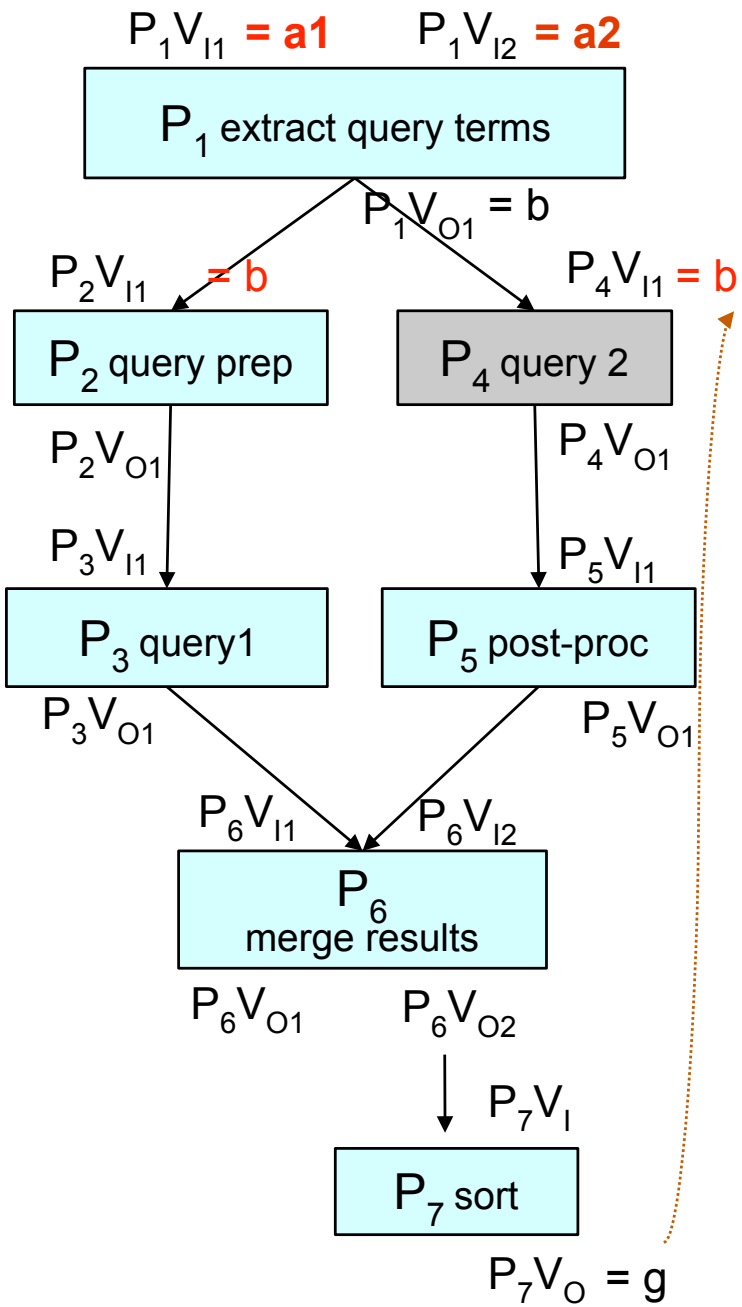
# Abstraction by selection



# Abstraction by selection



# Focusing – processor selection




| var | value |
|---|---|
| P1/VI1 | a1 |
| P1/VI2 | a2 |
| P1/VO1 | b |
| P2/VI1 | b |
| P4/VI1 | b |
| P2/VO1 | c1 |
| P4/VO1 | c2 |
| P7/VO | g |

P4 is the only interesting processor

Assume all values atomic

Query: lineage($P_7V_O$,{$P_4$})

Goal:

- avoid recursive queries on instance tables

Idea:

- use recursion on static model to generate a targeted query
- execute query only once

| xform() | | |
|---|---|---|
| IN | OUT | P |
| P4/VI1 | P4/VO1 | P4 |
| P5/VI1 | P5/VO1 | P5 |
| P6/VI2 | P6/VO2 | P6 |
| P7/VI | P7/VO | P7 |

| xfer() | |
|---|---|
| source | sink |
| P4/VO1 | P5/VI1 |
| P5/VO1 | P6/VI2 |
| P6/VO2 | P7/VI |

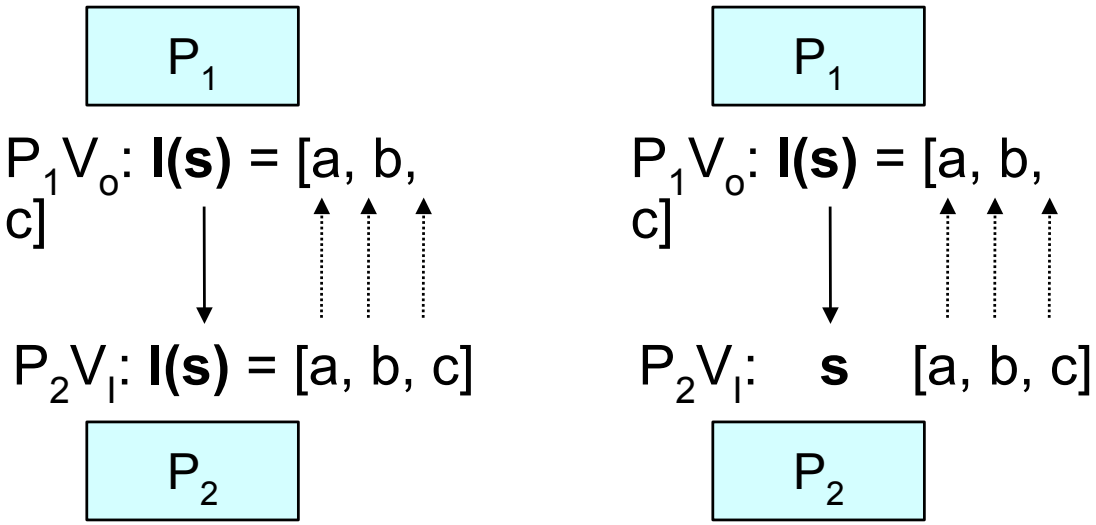
# Precision: elements within collections

## Problem: xform() also applies to list values

- It may be impossible to trace individual elements
  - "which pathways (out) depend on which genes **(in)**"?

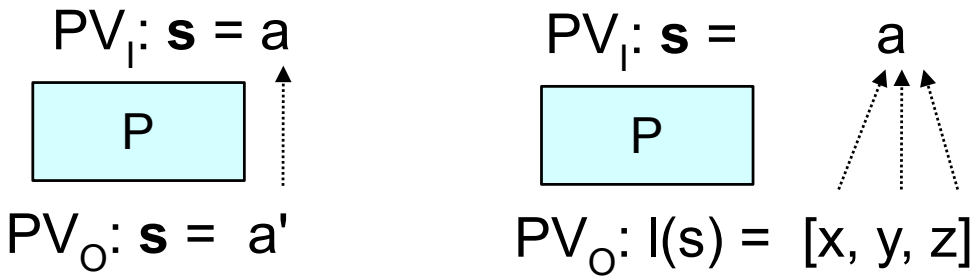
Goal: extend the query generation idea just sketched to trace element-level lineage within collections

Approach: exploit static typing of Taverna processors

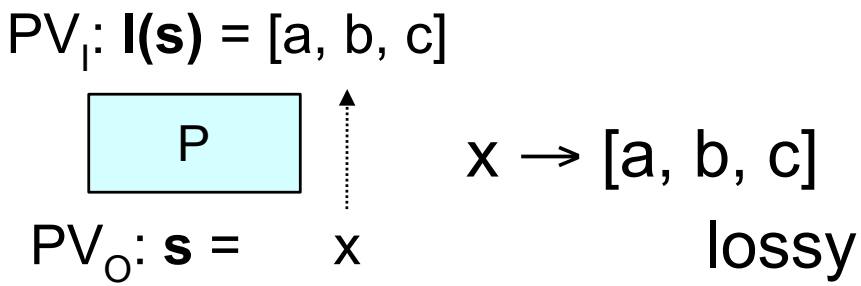


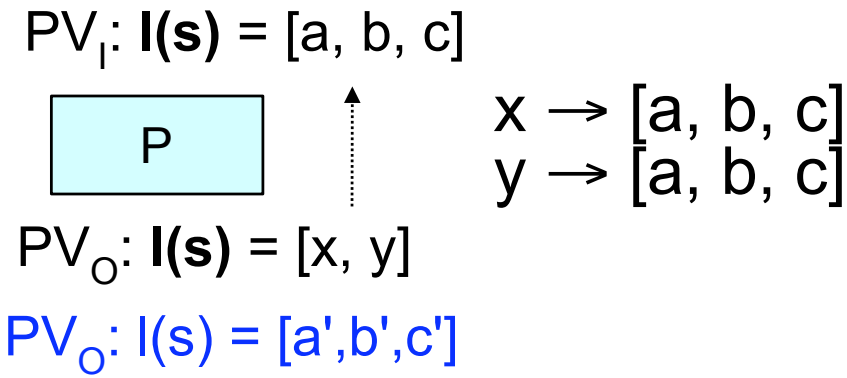

Taverna resolves mismatches on nesting levels: (map $P_2$ [a,b,c])

# Loss of precision in transformations




"lossless" transformations




possible behaviours:

- selection of an element
- **aggregation**
  **fun**
  **c**
  **tion f()** useful annotation:
  lineage(PV$_O$) = f(PV$_I$)

only useful annotation:
P is **index-preserving**:

$$PV_O[i] = PV_I[i]$$

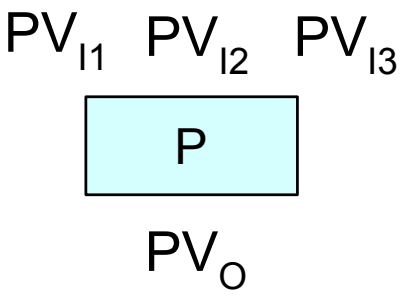
$$lineage(PV_O[i]) = PV_I[i]$$

# Cooperative processors

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

| | | |
|---|---|---|
| | $PV_I$: **I(s)** = [a, b, c]<br>P<br>$PV_O$: **s** = x | $PV_I$: **I(s)** = [a, b, c]<br>P<br>$PV_O$: **I(s)** = [x, y] |
| Static annotations: | aggregation f() | $PV_O[i] = PV_I[i]$ |
| Dynamic annotations: | selection:<br>$x = PV_I[i]$ | sorting:<br>$PV_O = \Pi(PV_I)$ |

# Other annotations

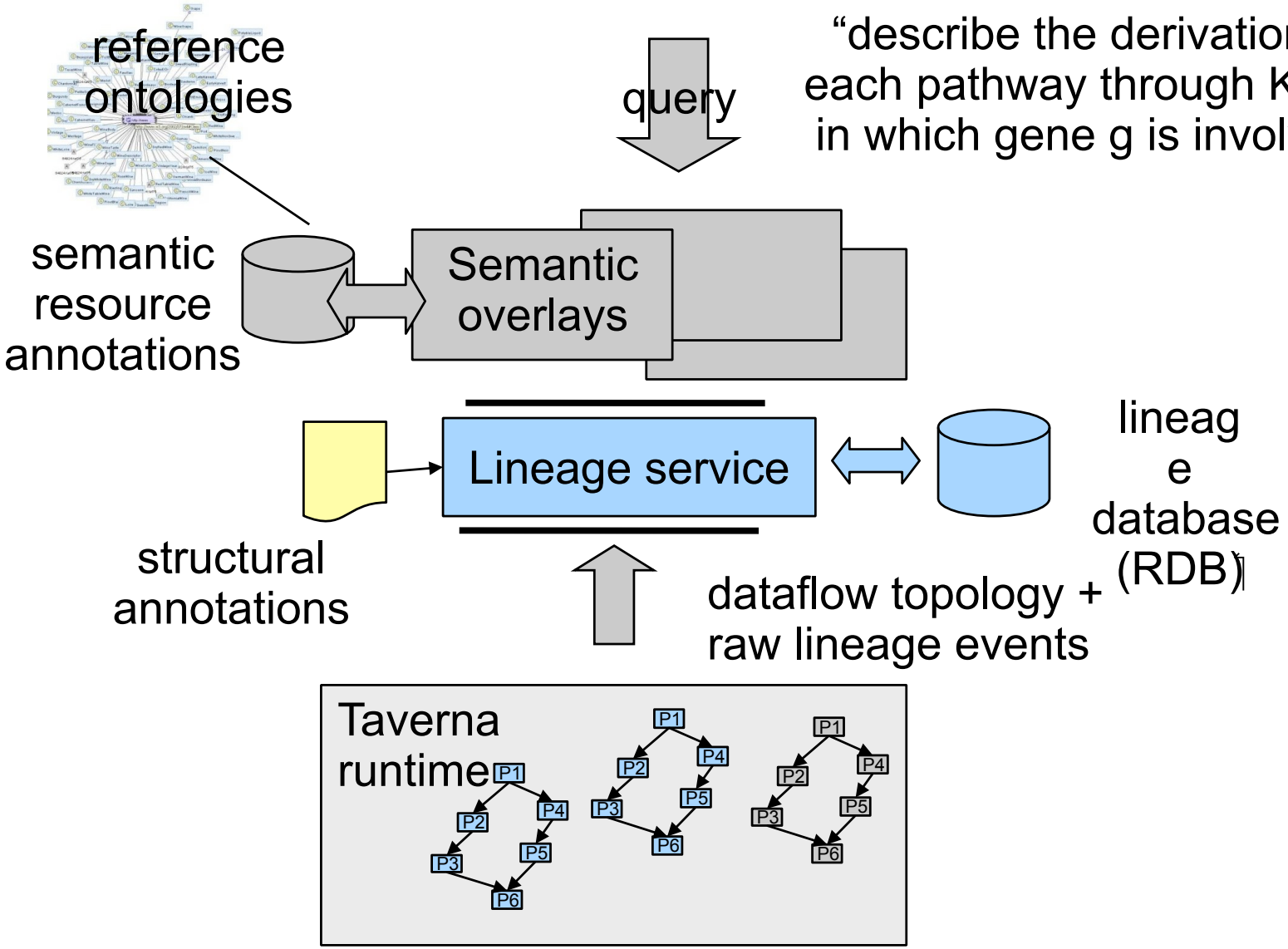



- Distinction between configuration and input data
  - $PVI_3$ is a configuration parameter
  - compare effect of different config. across multiple runs
- specific functional dependencies

  $[ PV_{I1}, PV_{I2} ] \rightarrow PV_O$
- stateless processor
  - execute process ↔ retrieve provenance

More evaluation needed on these

# Towards a 2 tier provenance model



# 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