Prospective and Retrospective Provenance Queries Using YesWorkflow, RDF, and SPARQL

Linh Hoang[1], Hui Lyu[2], Timothy McPhillips[1] and Bertram Ludäscher[1]

1. School of Information Sciences, University of Illinois at Urbana-Champaign 2. Department of Computer and Information Science, University of Pennsylvania

School of Information Sciences
The iSchool at Illinois

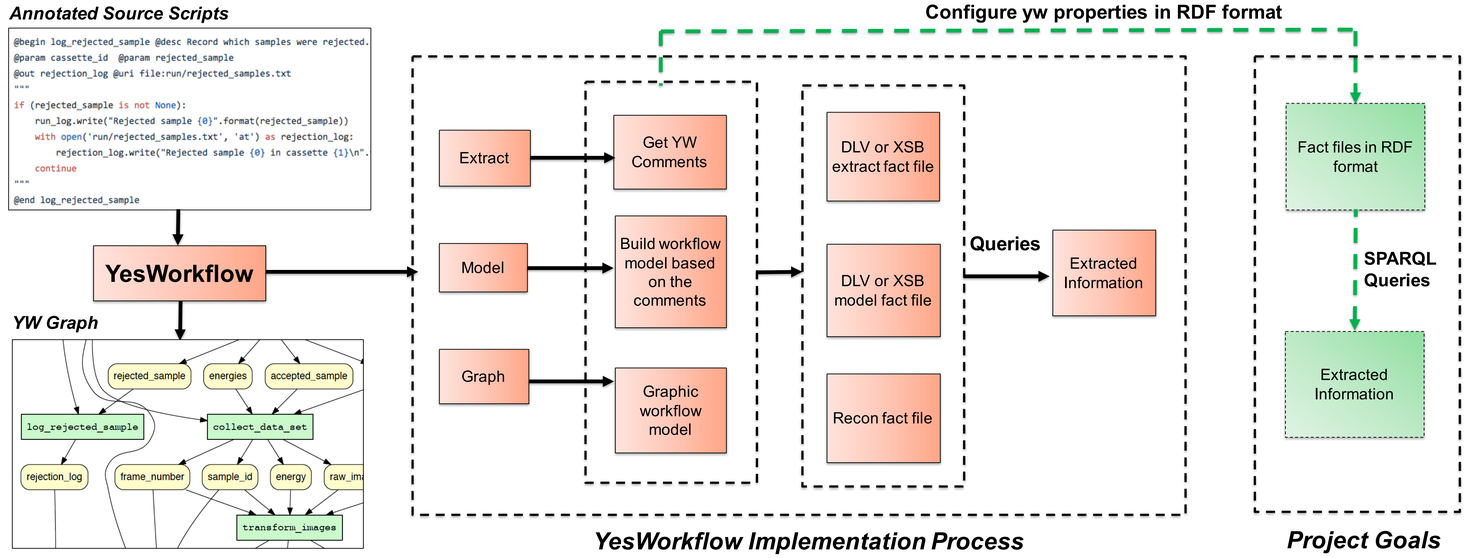
Motivation

DataONE

- Research projects in DataONE are using RDF to provide provenance from OAI-ORE* data packages.

YesWorkflow

- Defines a set of annotations for declaring the expected dataflow patterns in scripts written in any text-based programming language[1].
- YW builds a workflow model of the script based on these annotations, revealing the computational modules and dataflow dependencies in the script.
- YW can capture additional script-related artifacts, i.e., not only the workflow graph (*prospective* provenance) but also runtime observables (*retrospective* provenance) that can be queried by Prolog & Datalog.



Project Goals

- Allow YW-captured provenance information be easy to query together with other available provenance information which is already in RDF format.
- Enable all of the provenance information that can be collected by YesWorkflow and exported to Prolog facts, to be exported alternatively to an RDF representation[2].
- Produce RDF documents that are both easy to read directly and also easy to query using SPARQL.

Methods

(1) Conceptualized RDF model that captures the provenance related portions of YesWorkflow and created YW vocabularies.
(2) "Hand-crafted" RDF document based on the model for a specific YW example.
(3) Wrote SPARQL queries to mimic existing (working) Prolog / Datalog queries to ensure that the RDF model can support the same queries.
(4) Using two different SPARQL query engines (ARQ, Virtuoso) to test the queries.

References

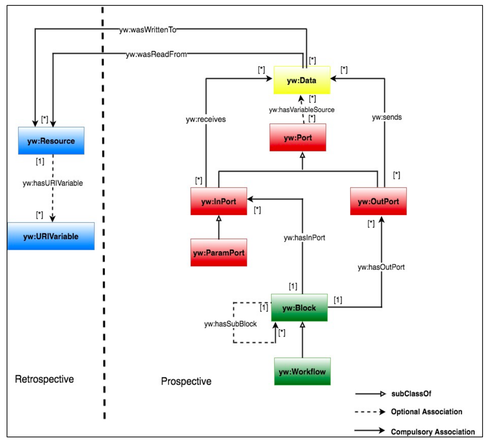
(1) McPhillips, T., Song, T., Kolisnik, T., Aulenbach, S., Belhajjame, K., Bocinsky, K., Cao, Y., Chirigati, F., Dey, S., Freire, J. and Huntzinger, D.: YesWorkflow: a user-oriented, language-independent tool for recovering workflow information from scripts (2015).
(2) DataONE Internships Summer 2017, https://www.dataone.org/internships#project3.
(3) The ProvONE Data Model for Scientific Workflow Provenance, http://jenkins-1.dataone.org/jenkins/view/Documentation%20Projects/job/ProvONE-Documentation-trunk/ws/provenance/ProvONE/v1/provone.html.

GitHub Repository

DataONE-Prov-Summer-2017 on GitHub,
https://github.com/idaks/DataONE-Prov-Summer-2017

DataONE
Data Observation Network for Earth

(1) YW RDF Model



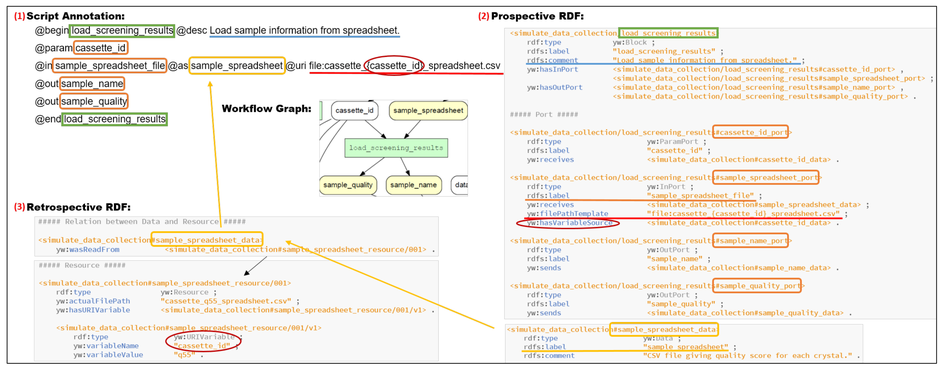
YesWorkflow RDF Model UML Diagram

- Was designed to integrate both prospective and retrospective provenance and was mapped with ProvONE* Model[3].

- **Prospective information:**
  - Block which represents for computational tasks in the script.
  - Port which represents for the connecting points in which blocks sending or receiving data to each other.
  - Data which represents for the data that flows between blocks via ports through out the script.

- **Retrospective information:**
  - Resource: which represents for physical instances of data generated during runtime events.
  - URIVariable: which represents for resource file name with its path.

(2) RDF Documents



Example of RDF representation & naming scheme

(3) (4) SPARQL Queries

Example 1: Recursive query using SPARQL property path
Question: What are the downstream blocks of the program block named "B1"?
Query:

```
SELECT DISTINCT ?downstream_block_name
WHERE
{ ?block          rdf:type          yw:Block;
                  rdfs:label        "B1".
  ?block  (yw:hasOutPort/yw:sends/^yw:receives/^yw:hasInPort)+  ?down_block.
  ?down_block     rdf:type     yw:Block;
                  rdfs:label   ?downstream_block_name.
} ORDER BY        ? downstream_block_name;
```

Example 2: Querying across between prospective and retrospective provenance[4]
Question: What URI variable value that associated with a data named "D1"?
Query:

```
SELECT DISTINCT ?variable_value
WHERE
{  ?data          rdf:type      yw:Data ;
                  rdfs:label    "D1" ;
                  (yw:wasReadFrom|yw:wasWrittenTo)     ?resource.
   ?resource      rdf:type yw:Resource;
                  yw:hasURIVariable      ?URIVariable .
   ?URIVariable   rdf:type              yw:URIVariable;
                  yw:variableValue       ?variable_value.}
```

Findings & Possible Future Work

- ARQ vs. Virtuoso: ARQ took longer time to run all the queries than Virtuoso.
- SPARQL property paths work nicely.
- Using standard RDF such as owl:sameAs, rdfs:subClassOf are not that helpful without reasoners.
- Examining the feasibility of YW RDF model, RDF representation & SPARQL queries with new examples.
- Defining "meaningful" queries that can be used by researchers.
- Judgments of using own RDF vocabulary instead of standard ProvONE vocabulary via real example.