A Rosetta Stone for Provenance Models

Michael R. Gryk, Pratik Shrivastava & Bertram Ludäscher
School of Information Sciences, University of Illinois at Urbana-Champaign, USA

ILLINOIS
School of Information Sciences

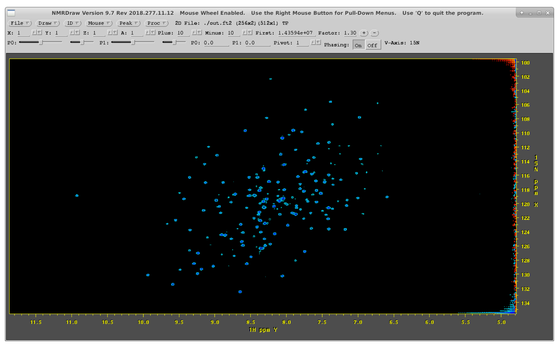
Abstract

With respect to scientific workflows, provenance refers to the documented lineage of how one dataset was produced from others. Provenance comes in at least two forms: retrospective provenance entails execution logs and provenance traces stored after a scientific workflow has been executed, and which describes the execution itself; prospective provenance refers to what a scientific workflow is designed to do, or in other words, prospective provenance describes the predicted lineage one would expect to have after a workflow has been executed. Not only are there various notions of provenance, there are also various models for tracking provenance. In this poster we compare and contrast four different provenance models: the prospective models of Common Workflow Language (CWL) and YesWorkflow, and the retrospective models of PROV and PREMIS. This comparison is made by documenting each of the various modelling constructs on the same workflow records – providing a Rosetta Stone for translating the provenance semantics between the various models.

Parent Shell Script

```
#! /bin/csh

nmrPipe -in hsqc.pipe \
| nmrPipe -fn ZF -size 2048 \
| nmrPipe -fn FT \
| nmrPipe -fn PS -p0 68.6 -p1 -34.8 -di \
| nmrPipe -fn EXT -left -sw \
| nmrPipe -fn TP \
| nmrPipe -fn LP -fb -ord 30 -x1 2 -xn 128
          -pred 64 -fix -fixMode -after \
| nmrPipe -fn ZF -size 256 \
| nmrPipe -fn FT \
-out hsqc.ft2 -ov
```

Output from Script



Common Workflow Language*

Workflow.cwl

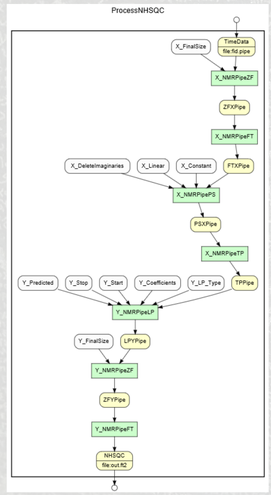
```
steps:

  xzf:
    run: zf.cwl
    in:
      zf_input: fid
      zf_size: xZF_size
      zf_output: xzf_outFile
    out: [zf_out]
  xft:
    run: ft.cwl
    in:
      ft_flag: xFT_flag
      ft_input: xzf/zf_out
      ft_output: xft_outFile
    out: [ft_out]
  xps:
    run: ps.cwl
    in:
      ps0: xps0
      ps1: xps1
      ps_input: xft/ft_out
      ps_output: xps_outFile
    out: [ps_out]
  xextract:
    run: ext.cwl
    in:
      ext_input: xps/ps_out
      ext_output: xext_outFile
    out: [ext_out]
  tp:
    run: tp.cwl
    in:
      tp_input: xextract/ext_out
      tp_output: tp_outFile
    out: [tp_out]
  ylp:
    run: lp.cwl
    in:
      lp_coeff: yLP_coeff
      lp_pred: yLP_pred
      lp_start: yLP_start
      lp_size: yLP_size
      lp_input: tp/tp_out
      lp_output: ylp_outFile
    out: [lp_out]
  yzf:
    run: zf.cwl
    in:
      zf_input: tp/tp_out
      zf_size: yZF_size
      zf_output: yzf_outFile
    out: [zf_out]
  yft:
    run: ft.cwl
    in:
      ft_flag: yFT_flag
      ft_input: yzf/zf_out
      ft_output: yft_outFile
    out: [ft_out]
```

Workflow.yml

```
fid:
  class: File
  path: hsqc.pipe
xZF_size: 1024
xzf_outFile: xzfout_temp
xFT_flag: TRUE
xft_outFile: xftout_temp
xps0: 68.6
xps1: -34.8
xps_outFile: xpsout_temp
xext_outFile: xextout_temp
tp_outFile: tpout_temp
yLP_coeff: 30
yLP_pred: 64
yLP_start: 2
yLP_size: 128
ylp_outFile: ylpout_temp
yZF_size: 256
yzf_outFile: yzfout_temp
yFT_flag: TRUE
yft_outFile: out.ft2
```

Yes Workflow

```
# @BEGIN ProcessNHSQC
# @IN TimeData @URI file:fid.pipe
#    @BEGIN X_NMRPipeZF
#    @IN TimeData @URI file:fid.pipe
#    @PARAM X_FinalSize
#    @OUT ZFXPipe
#    @END X_NMRPipeZF
#    @BEGIN X_NMRPipeFT
#    @IN ZFXPipe
#    @OUT FTXPipe
#    @END X_NMRPipeFT
#    @BEGIN X_NMRPipePS
#    @IN FTXPipe
#    @PARAM X_Constant
#    @PARAM X_Linear
#    @PARAM X_DeleteImaginaries
#    @OUT PSXPipe
#    @END X_NMRPipePS
#    @BEGIN X_NMRPipeTP
#    @IN PSXPipe
#    @OUT TPPipe
#    @END X_NMRPipeTP
#    @BEGIN Y_NMRPipeLP
#    @IN TPPipe
#    @PARAM Y_LP_Type
#    @PARAM Y_Coefficients
#    @PARAM Y_Start
#    @PARAM Y_Stop
#    @PARAM Y_Predicted
#    @OUT LPYPipe
#    @END Y_NMRPipeLP
#    @BEGIN Y_NMRPipeZF
#    @IN LPYPipe
#    @PARAM Y_FinalSize
#    @OUT ZFYPipe
#    @END Y_NMRPipeZF
#    @BEGIN Y_NMRPipeFT
#    @IN ZFYPipe
#    @OUT NHSQC @URI file:out.ft2
#    @END Y_NMRPipeFT
# @OUT NHSQC @URI file:out.ft2
# @END ProcessNHSQC
```



PREMIS* (* records are abbreviated for brevity)

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<premis:premis xmlns:premis="http://www.loc.gov/premis/v3"
   xmlns:builder="https://raw.githubusercontent.com/CONNJUR/CONNJUR_ML/master/connjur_ml.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="3.0"
   xsi:schemaLocation="http://www.loc.gov/premis/v3 http://www.loc.gov/standards/premis/premis.xsd">
  <premis:object xsi:type="premis:representation">
    <premis:objectIdentifier>
      <premis:objectIdentifierType>local</premis:objectIdentifierType>
      <premis:objectIdentifierValue>15N HSQC_001</premis:objectIdentifierValue>
    </premis:objectIdentifier>
    <premis:significantProperties>
      <premis:significantPropertiesType>format</premis:significantPropertiesType>
      <premis:significantPropertiesValue>NBX</premis:significantPropertiesValue>
    </premis:significantProperties>
  </premis:object>
  <premis:event>
    <premis:eventIdentifier>
      <premis:eventIdentifierType>local</premis:eventIdentifierType>
      <premis:eventIdentifierValue>Event_002</premis:eventIdentifierValue>
    </premis:eventIdentifier>
    <premis:eventType>nmrPipe -fn FT</premis:eventType>
    <premis:eventDateTime>Fri Aug 03 11:41:47 EDT 2018</premis:eventDateTime>
    <premis:linkingAgentIdentifier>
      <premis:linkingAgentIdentifierType>local</premis:linkingAgentIdentifierType>
      <premis:linkingAgentIdentifierValue>NMRPipe_001</premis:linkingAgentIdentifierValue>
    </premis:linkingAgentIdentifier>
    <premis:linkingObjectIdentifier>
      <premis:linkingObjectIdentifierType>local</premis:linkingObjectIdentifierType>
      <premis:linkingObjectIdentifierValue>15N HSQC_001</premis:linkingObjectIdentifierValue>
    </premis:linkingObjectIdentifier>
  </premis:event>
  <premis:agent>
    <premis:agentIdentifier>
      <premis:agentIdentifierType>local</premis:agentIdentifierType>
      <premis:agentIdentifierValue>NMRPipe_000</premis:agentIdentifierValue>
    </premis:agentIdentifier>
    <premis:agentName>NMRPipe</premis:agentName>
    <premis:agentType>software</premis:agentType>
    <premis:agentNote>9.2</premis:agentNote>
    <premis:linkingEventIdentifier>
      <premis:linkingEventIdentifierType>local</premis:linkingEventIdentifierType>
      <premis:linkingEventIdentifierValue>Event_002</premis:linkingEventIdentifierValue>
    </premis:linkingEventIdentifier>
  </premis:agent>
```

PROV

PROV	PREMIS	CWL	YesWorkflow
Entity	Object	File	Port
Plan	N/A	Workflow	Top Level
Agent	Agent	BaseCommand	N/A
Activity	Event	CommandLineTool	Actor

References & Acknowledgemnts

- Amstutz, P. Michael R. Crusoe, Nebojša Tijanić (editors), Brad Chapman, John Chilton, Michael Heuer, Andrey Kartashov, Dan Leehr, Hervé Ménager, Maya Nedeljkovich, Matt Scales, Stian Soiland-Reyes, Luka Stojanovic (2016): Common Workflow Language, v1.0. Specification, Common Workflow Language working group.
- McPhillips, et al. (2015). "YesWorkflow: A User-Oriented, Language-Independent Tool for Recovering Workflow Infor-mation from Scripts. International Journal of Digital Curation 10, 298-313.
- Moreau, L., Missier, P.; eds. (2013) PROV-DM: The PROV Data Model. 30 April 2013, W3C Recommendation. URL: http://www.w3.org/TR/2013/REC-prov-dm-20130430/
- PREMIS Editorial Committee (2015) PREMIS Data Dictionary for Preservation Metadata version 3.0. Library of Congress, Washington, DC.
- Stodden V, McNutt M, Bailey DH, Deelman E, Gil Y, Hanson B, Heroux MA, Ioannidis JPA, Taufer M. (2016). Enhancing reproducibility for computational methods. Science 354, 1240–1241.
- Rosetta Stone photo: © Hans Hillewaert, CC BY-SA 4.0

The authors wish to thank the iSchool Center for Informatics Research in Science and Scholarship for support.

Conclusions and Lessons Learned

• YesWorkflow explicitly labels data inputs and outputs (implied by pipes in shell script). YW decomposes parameters from processes.

• CWL decomposes the configuration of an individual function from the steps of the workflow. Also sets parameters in a YAML file. Decomposition facilitates the reuse of "zf.cwl" and "ft.cwl"

• PREMIS records details of "OBJECT" (dataset in CWL/YW), "EVENT" (actor/tool in CWL/YW) and "AGENT" (not defined in YW, "baseCommand" in CWL).

• PROV also has notion of "ENTITY" (PREMIS:OBJECT), "ACTIVITY" (PREMIS:EVENT) and "AGENT". In addition, PROV allows finer grained decomposition: ACTIVITY is conducted by a SOFTWARE AGENT at the bequest of a human AGENT. In addition, data dependencies and process dependencies use richer terminology to span larger portions of the graph.