Representing Microarray Experiment Metadata Using Provenance Models

James McCusker

February 9, 2010

Abstract

MAGE (MicroArray and Gene Expression) representations are primarily representations of workflow: a process was used to derive biomaterial A from biomaterial B. This representation is ideally suited for representation using provenance models such as OPM (Open Provenance Model) and PML (Proof Markup Language). We demonstrate methods and tools, MAGE2OPM and MAGE2PML, to convert RDF representations of RDF MAGE graphs to OPM and PML respectively. We evaluate the representations' expressiveness, visualization, and integration with existing data. We argue that provenance models are sufficient to represent biomedical experimental metadata in general, and may provide a useful point of reference for unifying information from multiple systems, including biospecimen management, Laboratory Information Systems (LIMS), and computational workflow automation tools. This unification results in a complete, computationally useful derivational picture of biomedical experimental data.

1 Introduction

Provenance, as a concept, is critical for successful scientific research. As such, it is vital that systems that support the sciences provide a framework for incorporating provenance information at every step of the research chain. Further, the context that is provided by richly encoded provenance can be used to automate certain aspects of scientific research. For instance, if the system were able to tell that the samples used in two different experiments were derived from the same source (such as a tumor, blood sample, or patient) it could possibly automatically integrate those datasets at the level of specificity that the researcher needs. By converting current representations of experimental metadata, such as use of protocols, experimental conditions, which samples were derived in what ways, etc. into a unified model of provenance, it becomes possible to support automated data integration and experimental workflow analysis. A first step in this process is to convert experimental data into a common provenance representation.
MAGE (MicroArray and Gene Expression) representations are primarily representations of experimental workflow: a process was used to derive biomaterial A from biomaterial B. This representation is ideally suited for representation using provenance models such as OPM (Open Provenance Model) [3] and PML (Proof Markup Language) [1]. I argue that provenance models are sufficient to represent biomedical experimental metadata in general, and may provide a useful point of reference for unifying information from multiple systems, including biospecimen management, Laboratory Information Systems (LIMS), and computational workflow automation tools. This unification results in a complete, computationally useful derivational picture of biomedical experimental data.

2 Related Work

- Taverna and WINGS provenance recording
- OPM
- PML
- MGED and MAGE

3 Methods

3.1 MAGE-RDF to Open Provenance Model

- Subclasses
  - opm:Account
    * mged:Experiment
  - opm:Agent& subproperties
    * mged:Contact
  - opm:Artifact
    * mage:ArrayHybridization
    * mage:ParameterValue
    * mged:ArrayGroup
    * mged:BioAssay
    * mged:BioAssayData
    * mged:BioMaterial
    * mged:ExperimentalDesign
    * Observation
  - opm:Process
    * mged:Protocol
14 SWRL Rules (using swrlx:makeOWLThing())

1. opm:Derived(?derived) ^
   opmd:derivedArtifact(?derived, ?node) ^
   mged:has_nodes(?exp, ?node)
   =>
   opmd:derivedAccount(?exp, ?derived)

2. mged:has_derivative(?node, ?source) ^
   swrlx:makeOWLThing(?derived, ?node, ?source)
   =>
   opmd:derivedArtifact(?derived, ?source) ^
   opmd:derivedFromArtifact(?derived, ?node)

3. maged:ProtocolApplication(?pa) ^
   mged:has_nodes(?exp, ?pa) ^
   maged:has_parameter_value(?pa, ?pv) ^
   maged:has_parameter(?pv, ?param) ^
   maged:has_parameter(?protocol, ?param) ^
   swrlx:makeOWLThing(?used, ?pv)
   =>
   opm:Used(?used) ^
   opm:Artifact(?pv) ^
   opm:usedArtifact(?used, ?pv) ^
   opm:usedRole(?used, ?param) ^
   opm:usedAccount(?used, ?exp) ^
   opm:usedByProcess(?used, ?protocol)

4. opm:Controlled(?pa) ^
   mged:has_performer(?pa, ?performer)
   =>
   opm:controlledByAgent(?pa, ?performer)

5. opm:Controlled(?pa) ^
   opm:controlledByAgent(?pa, ?performer) ^
   mged:has_type(?performer, ?role) ^
   mged:Roles(?role)


- MGED Ontology:Experiment(?exp) ^ mged-has_nodes(?exp, ?a) ^ MGED Ontology:has_biomaterial_characteristics(?a, ?c) ^ swrlx:makeOWLThing(?derived, ?c) -> opm:Derived(?derived) ^ opm:derivedFromArtifact(?derived, ?a) ^ opm:derivedArtifact(?derived, ?c) ^ opm:derivedAccount(?derived, ?exp)

- opm:controlled(?pa) ^ MGED Ontology:has_protocol(?pa, ?protocol) -> opm:controlledProcess(?pa, ?protocol)

- opm:generatedArtifact(?generated, ?artifact) ^ has_default_role(?artifact, ?role) -> opm:generatedRole(?generated, ?role)

- opm:usedArtifact(?used, ?artifact) ^ has_default_role(?artifact, ?role) -> opm:usedRole(?used, ?role)

• Roles mapped through restrictions on original classes.

3.2 MAGE-RDF to Proof Markup Language

• 3 SWRL Rules (using swrlx:makeOWLThing())

  - MAGE Node NodeSets:
    mged:Experiment(?exp) ^ mged-has_nodes(?exp, ?node) ^

•
\[
\text{swrlx:makeOWLThing(?nodeSet, ?node)} \quad ^{\rightarrow} \\
\text{pmlj:NodeSet(?nodeSet)} \quad ^{\rightarrow} \\
\text{pmlj:hasConclusion(?nodeSet, ?node)} \quad ^{\rightarrow} \\
\text{pmlj:isConsequentOf(?nodeSet, ?inferenceStep)} \\
\text{pmlj:InferenceStep(?inferenceStep)} \\
\text{pmlj:hasIndex(?inferenceStep, 0)} \\
\text{pmlj:hasSourceUsage(?inferenceStep, ?sourceUsage)} \\
\text{pmlj:hasSource(?sourceUsage, ?exp)}
\]

Derivation along ProtocolApplications:
\[
\text{mage:ProtocolApplication(?pa)} \quad ^{\rightarrow} \\
\text{mage:has derivation_source(?pa, ?derivation)} \\
\text{mged:has protocol(?pa, ?protocol)} \\
\text{mage:has derivative(?pa, ?derived)} \\
\text{mged:has protocol(?pa, ?protocol)} \\
\text{pmlj:hasConclusion(?derivedNodeSet, ?derived)} \\
\text{pmlj:hasConclusion(?derivationNodeSet, ?derivation)} \\
\text{swrlx:makeOWLThing(?antecedentList, ?pa)} \\
\]

FactorValues:
\[
\text{mged:FactorValue(?factorValue)} \\
\text{mged:has factor_value(?info, ?factorValue)} \\
\text{mged:has experiment_factors(?factorValue, ?factor)} \\
\text{mged:ExperimentalFactor(?factor)} \\
\text{pmlj:hasConclusion(?nodeSet, ?info)} \\
\text{swrlx:makeOWLThing(?tvNodeSet, ?factorValue)} \\
\text{swrlx:makeOWLThing(?inferenceStep, ?factorValue)} \\
\text{swrlx:makeOWLThing(?nodeSetList, ?factorValue)} \\
\]

\[
\text{pmlj:InferenceStep(?inferenceStep)} \\
\text{pmlj:NodeSetList(?nodeSetList)} \\
\text{pmlj:NodeSet(?tvNodeSet)} \\
\text{pmlj:hasConclusion(?tvNodeSet, ?factorValue)} \\
\text{pmlj:isConsequentOf(?tvNodeSet, ?inferenceStep)} \\
\text{pmlj:hasIndex(?inferenceStep, 0)} \\
\text{pmlj:hasInferenceRule(?inferenceStep, ?factor)}
\]
3.3 Evaluation

**Expressiveness & Coverage**: evaluate the number of Individuals that are connected to the provenance graph using the properties that either are part of the provenance model or subproperties of the provenance model.

**Visualization**: Display an example provenance graph using tools specifically built for each provenance model. Measure the information density of the visualization.

**Data Integration**: List and demonstrate additional graphs of provenance that can be added to the graph to extend the existing provenance graph longitudinally.

4 Results and Discussion

4.1 Expressiveness and Coverage

4.2 Visualization Tools

4.3 Data Integration

5 Conclusion