Validation of E-Science Experiments using a Provenance-based Approach

Sylvia Wong, Simon Miles, Weijian Fang, Paul Groth and Luc Moreau

University of Southampton, UK
Overview

- E-Science experiment validation
- Bioinformatics scenario
- Provenance-based validation architecture
- Evaluation results
E-Science Experiments

- Large scale computations for conducting scientific research
- Multiple distributed services on the Grid
- Workflow validation
  - Part of the scientific process
  - Verify correctness of their own experiments
  - Review correctness of their peers' work
Static Validation

- Operates on workflow source code
- Checks if workflow satisfies some properties before it is run
- Examples
  - type inference, escape analysis, concurrency analysis, graph-based partitioning
- Workflow script may not be accessible or may be expressed in a language not supported by analysis tool
Dynamic Validation

- Verifies data values satisfy constraints during execution
  - interface matching, runtime type checking
- Cannot assume services will perform validation
- Interfaces may be under-specified
  - In bioinformatics, biological sequences commonly specified as strings in interfaces
Provenance-based Validation

- Allows for validation of experiments after execution
- Third parties may want to verify that the results obtained were computed correctly according to some criteria
- These criteria may not be known when the experiment was designed or run
- Important because science progresses (and models evolve!)
Bioinformatics Scenario

- A biologist has a set of proteins, for each of which he/she wishes to determine a particular biological property.
Experiment Services

- Design experiment (abstract plan)
- For each step in the plan, decide on the concrete service to use
- Each service may be designed by the biologist or adopted from the work of another biologist
- For each service there is a description of that service stating:
  - what the service does
  - what type of data it analyses (its inputs) and
  - what type of results it produces (its outputs)
- All the descriptions are stored in a registry

<table>
<thead>
<tr>
<th>Service A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function: .....</td>
</tr>
<tr>
<td>Inputs: .....</td>
</tr>
<tr>
<td>Outputs: .....</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service B</th>
</tr>
</thead>
<tbody>
<tr>
<td>............</td>
</tr>
<tr>
<td>............</td>
</tr>
<tr>
<td>............</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service C</th>
</tr>
</thead>
<tbody>
<tr>
<td>............</td>
</tr>
<tr>
<td>............</td>
</tr>
<tr>
<td>............</td>
</tr>
<tr>
<td>............</td>
</tr>
</tbody>
</table>

Registry
Performing Experiment

- Performs experiment
- Details of experimental process documented in a provenance store
- Each service documenting its own execution
Questions

1. Did I perform each service on the type of data that the service was intended to analyse?
   • Were the inputs and outputs of each activity compatible?

2. Did the services I used actually fulfil my high level plan?
Answering the Questions

- Using the documentation in the Provenance Store, we can reconstruct the process that led to each result.
- Along with the high level plan and the descriptions in the registry we have all the information required to answer the questions.
Q1: Were the inputs and outputs compatible?

- **Provenance Store**
  - Retrieve each pair of services performed in an experiment, where one service’s output is the other’s input

- **Registry**
  - Retrieve descriptions for each service

- **Description of Service A**
  - Function: …..
  - Inputs: …..
  - Outputs: …..

- **Description of Service B**
  - Function: …..
  - Inputs: …..
  - Outputs: …..

Compare the output type of the first service with the input type of the second
Q2: Did the experiment follow the plan?

Retrieve documentation of experiment that led to a result

Retrieve procedure descriptions

Description of Service A
Function: ..... Inputs: ..... Outputs: ..... Compare procedure function to planned activity

Provenance Store

Registry
High-level activity may be described in a more general way than the service which performs it.

Also, one service’s input may be a generalisation of the preceding service’s output.

Therefore, exact matching of types may produce a false negative: the biologist will wrongly be told the experiment was invalid.

By using an ontology, describing how types are related, we can reason about types and determine whether they are truly compatible.
Architecture
Testing

- Workflow - protein compressibility
- Provenance store – PASOA (pasoa.org)
- Registry – Grimoires (grimoires.org)
- Validator – Java, Jena 2.1
- Ontology in OWL, based on myGrid bioinformatics ontology
Performance Evaluation

- Potentially, large number of experiments are performed
- Evaluate if our approach can scale with the size of the provenance store
- Time to validate an experiment with increasing number of experiments recorded
Performance

input/output type validation

plan validation
Summary

- Provenance-based validation of workflow executions
  - Validation of experiments after execution
    - Previously unknown criteria
    - Third party validation
- Tested with a sample bioinformatics experiment
- Evaluation shows framework scales well with increasing data store size