From Scripts towards Provenance Inference

Rezwanul Huq*, Peter M.G. Apers, Andreas Wombacher
University of Twente, The Netherlands.

Yoshihide Wada, Ludovicus P. H. van Beek
Utrecht University, The Netherlands.
eScience Application

Workflow
Workflow Model: Activity

- **Views**: data
- **Interval predicates**: \( \sigma_{I1}(V_1) \), \( \sigma_{I2}(V_2) \), \( \sigma_{I3}(V_3) \)
- **Windows**
- **Trigger**: based on windows
- **Exactly one output view**
Workflow & Provenance

Why & Where?

- Provenance: derivation history of data products starting from its original sources.
Workflow Provenance Capture: State-of-art

Provenance-unaware Platform

Languages like Python
Tools like Excel, R

Provenance-aware Platform

Kepler, Taverna, Karma, VisTrails
STREAM, Aurora

Bridging Gap – How?

Manually Building

Time, Training

Workflow Provenance
Problem Statement

How to capture Workflow Provenance automatically in a provenance-unaware platform?
Our Contribution: Workflow Provenance Inference

Provenance-unaware Platform
- Languages like Python
- Tools like Excel, R

Provenance-aware Platform
- Kepler, Taverna, Karma, VisTrails
- STREAM, Aurora

Workflow Provenance Inference

Workflow Provenance

For Python

UNIVERSITY OF TWENTE.

eScience 2012
Workflow Provenance Inference: Challenge

- Capturing *data dependences* by analyzing the script.
- Translating control dependences into data dependences.

```python
if DEBUG:
    value = -1
else:
    value = 1
print value
```
Workflow Provenance Inference: Overview

Python Script → Parsing\(^1\) → AST

Transformation\(^2\) → Objects → Traversing

Initial Graph → Re-writing\(^2\) → Provenance Graph

\(^1\) off-the-shelf grammar from ANTLR site
\(^2\) Attributed Graph Grammar (AGG)
Provenance Graph Model

- Represented as a graph \(\rightarrow\) Provenance graph

- **Constant**
  - ID
  - Value
  - Type
  - Line#

- **Source Processing Element**
  - ID
  - Name
  - Type
  - Start Line#
  - End Line#
  - windows
  - trigger
  - input-output ratio

- **Computing Processing Element**
  - ID
  - Name
  - Type
  - Start Line#
  - End Line#
  - hasOutput
  - windows
  - trigger
  - input-output ratio

- **View**
  - ID
  - Name
  - IsPersistent
  - IsIntermediate
  - Line#

- \(\rightarrow\) Windows
- \(\rightarrow\) Trigger
- \(\rightarrow\) Input-output ratio
- \(\rightarrow\) hasOutput
Transformation Phase

- Building the initial graph
- Preserving order between statements
- Maintaining versions of variables
Transformation: An example

2 \( \text{var1} = 100 \)
3 \( \text{var3} = \text{var1} + 10 / 50 \)
4 \( \text{var1} = \text{var3} \)
Re-writing Phase

- A rule consists of \textit{LHS} and \textit{RHS}.
- A pattern matches to \textit{LHS} will be replaced by \textit{RHS}.

- Re-write rules for:
  - Translating \textit{control-flow} statements
  - Maintaining persistence of views
  - Ensuring compactness of the graph
Re-writing: An example

11 for i in range(start,end,incr):
12     value = ...

**Rule: LHS**

- C₁:start
- C₂:end
- C₃:incr

P₁: range → V₁ → P₂: forLoop → V₂ → Processing chain → V₃

Loop control variable: i

**RHS**

- C₁:start
- C₂:end
- C₃:incr

P₁: range → V₁ → Processing chain → V₃

Windows = end-start+1
Trigger = incr
Workflow Provenance Graph

Before Re-writing

**C₁**: 100 (#2) → **P₁**: = (#2) → **V₁**: var1 (#2)

**C₂**: 10 (#3)

**C₃**: 50 (#3) → **P₂**: / (#3) → **V₂**: int. (#3)

After Re-writing

**C₁**: 100

**P₁**: = (#2) → **V₁**: var1

**C₂**: 10

**P₂**: / (#3) → **V₂**: int.

**C₃**: 50

**P₃**: + (#3) → **V₃**: int. (#3)

**P₄**: = (#3) → **V₄**: var3

**P₅**: = (#4) → **V₅**: var1
Evaluation: Use Case

- Water Scarcity Modeling.
- We focus on estimating **irrigation water demand**.
- Several files (PCRaster maps with 360*720 dimension) are used with different PCRaster operations.
Evaluation: Quantitative Analysis

Workflow Provenance

Initial Graph: ~ 450 nodes

Final Graph: ~ 139 nodes

Fine-grained Provenance Inference

> 3000 maps

~ 40 GB offline data

Inference Methods
Evaluation: Qualitative Analysis (I)

- Open-ended interview with two scientists
  - Debugging-friendliness
  - Extensibility
  - Customization
Evaluation: Quantitative Analysis (II)

“Sometimes, I used to spend hours finding reasons for having an unexpected value.”

**Extensibility**

→ Need to enter few information for the very first run.

“This is too detailed. I want to group some elements to have an overview of the processing.”

**Debugging-friendliness**

→ Easy access to data
→ code efficiency

“I need to access library functions or functions written elsewhere.”

**Customization**

→ Adaptation based on user preference is possible.
Conclusion & Future Plan

- Workflow provenance capture in provenance-unaware platform
  - Manually capturing requires both time, training

Workflow Provenance Inference

Future Plan
- Address other control-flow statements
- Build a complete framework with GUI
- Poster in AGU Fall Meeting 2012 at San Francisco
Thank You 😊