Extending the PROV Model to Data Transformation Scripts

George Alter
SDTH: Structured Data Transformation History

• SDTH answers basic questions about a data transformation script:
  • What dataframes/variables affected the values of variable X or dataframe Y?
  • What dataframes/variables were affected by variable X or dataframe Y?
  • What commands affected the values of variable X or dataframe Y?
  • What commands were affected by variable X or dataframe Y?

• SDTH is an extension of the W3C PROV standard for provenance
  • Expressed as RDF triples (subject, predicate, object)
  • Can be queried with SPARQL
  • Can be combined with other PROV statements
W3C PROV

• Key concepts in PROV are very abstract
  • Activity
  • Entity
  • Activities can use an Entity and can generate a new Entity

• Focuses on origin, ownership, and **process**
  • PROV is a history of data flowing through processes
  • A process has inputs and outputs
  • Operations within a process are not described
W3C PROV: Limitations

• A PROV process is a black box
  • PROV activities are only described by inputs and outputs
  • PROV entities do not change.
  • Activities create new entities

• PROV does not meaningfully describe data sets
  • No concept corresponding to a data matrix (e.g., dataframe)
  • No way to represent a variable or record within a data structure

• SDTH extends PROV to data sets and data transformations
Variable names vs. VariableInstances

- Variables are not PROV entities
  - PROV entities are stable and immutable
  - Variables in procedural languages (SPSS, R, etc.) change all the time
    - A variable is a container with a variable name
    - Contents of a variable may change, but the variable name remains the same
- SDTH introduces instances
  - An instance is a specific state of a data structure (variable, dataframe, file)
  - A VariableInstance is a specific set of values
  - If a value in a variable changes, a new VariableInstance is created
  - A variable name may be associated with many VariableInstances
Names and Instances: A Simple Example

• A simple data transformation script:
  
  Compute varX = 10
  Compute varX = 2*varX

• One variable **name**: “varX”

• Two variable **instances**: 10, 20
Entities and Predicates in SDTH

- RDF consists of triples: (Subject, Predicate, Object)
  - Subject and Object must be PROV entities
  - Predicates are PROV activities or attributes of entities

- An SDTH Program has ProgramSteps
- An SDTH ProgramStep
  - Acts upon data instances
    - FileInstance
    - DataframeInstance
    - VariableInstance
  - Has source code

- A data instance
  - Has a name in the source code
  - May be derived from a previous data instance
    - derivedFrom is used when values change
  - May be elaborated from a previous data instance
    - elaboratedFrom is used when attributes of variables change
    - E.g., a numeric variable is converted to string
Example: Two Python Commands

```python
PersonalData = PersonalData.assign(HHsize=PersonalData['PPHHSIZE'])
```

- Adds a new variable named ‘HHsize’ to dataframe ‘PersonalData’
- Assigns values from ‘PPHHSIZE’ to ‘HHsize’

```python
PersonalData['HHcateg'] = pd.cut(PersonalData['HHsize'],
                                  [1, 2, 3, 5, 7, 10, 999],
                                  include_lowest=True, right=False,
                                  labels=['1', '2', '3-4', '5-6', '7-9', '10+']
```

- Adds a new variable named ‘HHcateg’ to dataframe ‘PersonalData’
- Assigns values from ‘HHsize’ to ‘HHcateg’
- Recodes ‘HHcateg’ into six categories
Graph of SDTH for Example

Entities are assigned identifiers

```
"PersonalData = PersonalData.assign(HHsize=PersonalData['PPHHSIZE'])"

"PersonalData['HHcateg'] = pd.cut(PersonalData['HHsize'], [1, 2, 3, 5, 7, 10, 999], include_lowest=True, right=False, labels=['1', '2', '3-4', '5-6', '7-9', '10+'])"
```
Graph of SDTH for Example

Source code is an attribute of a ProgramStep

```
"PersonalData = PersonalData.assign(HHsize=PersonalData['PPHHSIZE'])"

"PersonalData['HHcateg'] = pd.cut(PersonalData['HHsize'], [1, 2, 3, 5, 7, 10, 999], include_lowest=True, right=False, labels=['1', '2', '3-4', '5-6', '7-9', '10+'])"
```
A variable name is an attribute of a VariableInstance.
Graph of SDTH for Example

ProgramSteps

use and assign VariableInstances

"PPHHSIZE"

hasName

"HHsize"

hasName

"HHcateg"

hasName

#VarIns010

usesVariable

#ProgStep004

hasSourceCode

"PersonalData = PersonalData.assign(HHsize=PPHHSIZE)"

#VarIns013

assignsVariable

#ProgStep005a

hasSourceCode

"PersonalData['HHcateg'] = pd.cut(PersonalData['HHsize'], [1, 2, 3, 5, 7, 10, 999], include_lowest=True, right=False, labels=['1', '2', '3-4', '5-6', '7-9', '10+'])"
Graph of SDTH for Example

derivedFrom shows the origin of a VariableInstance
SPARQL: What variables affected HHcateg?

A simple SPARQL query gives us the names of all variables that affected ‘HHcateg’, directly or indirectly.

```
SELECT distinct ?sname ?oname
WHERE {
    ?s sdth:wasDerivedFrom+ ?o .
    ?s sdth:hasName ?sname .
    ?o sdth:hasName ?oname .
FILTER (?sname = "HHcateg")
}
```

Output of a SPARQL Query

<table>
<thead>
<tr>
<th>Subject variable name (?sname)</th>
<th>Object variable name (?oname)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HHcateg</td>
<td>HHsize</td>
</tr>
<tr>
<td>HHcateg</td>
<td>PPHHSIZE</td>
</tr>
</tbody>
</table>

SPARQL queries operate on VariableInstances, but they report variable names.
SDTH is a complement of SDTL

- Our research group also developed SDTL
- SDTL and SDTH solve different aspects of a common problem
  - How do we describe the ways that data have been transformed and manipulated?

**SDTL: Structured Data Transformation Language**
- SDTL is a language for describing data transformations by statistical analysis software
- SDTL was created to add variable-level provenance to metadata
- Many different languages can be translated into SDTL
- SDTL is machine-actionable (e.g., JSON), which simplifies writing software tools

**SDTH: Structured Data Transformation History**
- SDTH is a simple way to query scripts
- SDTH is much less detailed than SDTL
- SDTH extends the W3C PROV standard to describe data
  - SDTH can be combined with other PROV compatible provenance metadata
SDTH: Conclusions

• SDTH answers four basic questions about data transformation scripts
• By moving to the PROV model, SDTH makes querying much easier
• SDTH extends the PROV model with data objects
• PROV and SDTH require a change in perspective
  • From variable names, which point to containers for values
  • To variable instances, which are specific sets of values
How will SDTL and SDTH be maintained?

• DDI Alliance is a standards organization that maintains DDI and related standards
• SDTL has been adopted by the DDI Alliance
• SDTH is being documented for review by the DDI Alliance
• All software developed by the C²Metadata Project is open source on Gitlab
Reference links

• **Publications:**

• **SDTL product page:** https://ddialliance.org/products/sdtl
• **SDTL User Guide:** http://c2metadata.gitlab.io/sdtl-docs/master/
• **SDTL Working Group:** https://ddi-alliance.atlassian.net/wiki/spaces/DDI4/pages/899547182/SDTL+-+Structured+Data+Transformation+Language
Thank You!

Supported by National Science Foundation grant NSF ACI-1640575