Data and Workflow Provenance: An Overview

Shridevi Erayya Hombal

Data Analytics, IIIT Bangalore, Bangalore, 560100, India

Abstract -Provenance can be applicable for database and workflow management systems. In concern to databases the provenance has been defined for complex or relational data, by propagating fine-grained annotations or algebraic expressions from input to the output. Not only in the database the provenance is considered but also it can be applicable to many other areas of computer science such as probabilistic databases, schema and data integration, annotation databases, etc. Whereas, workflow provenance is crucial to verification of scientific computation which aims to capture a complete description of evaluation or enactment of a workflow. Graphical notation is used to represent the workflow and their provenance but basically it complicates the formal semantics that relates run time behaviour with the provenance records. This problem can be resolved by extending the previously developed dataflow language which supports both workflow-style and database-style provenance graph that can be explicitly queried.

Keywords: dataflow language, workflow, database, provenance

I. INTRODUCTION

Several standard database models include lineage[1], where provenance[2,3],why provenance[3,4] and more recent innovations such as dependency provenance[5], provenance traces[6], how provenance[7,8].

Similar to database provenance models, provenance models can also been developed for a variety of workflow systems, such as Kepler[9], Karma[10], Taverna[20], Chimera[10], and ZOOM[11]. These systems model and record provenance as a directed acyclic graph that, informally, describes the macroscopic computation steps performed in constructing intermediate and final results. Recently the Open Provenance Model(OPM) [12] has been developed as a consensus exchange format for representing provenance graphs. Workflow systems employ much variety of programming constructs including concurrency, procedures, service calls and queries to external databases.

Workflow systems are accompanied by semantics, with or without provenance. As a result it can be hard to understand provenance information produced by a workflow system. It is difficult to integrate database and workflow provenance or compare provenance graphs generated by different systems because of clear specification of the semantics and provenance behaviour. Hence it is essential to study the semantics of workflow provenance models and relate them to existing models of database provenance.

Database provenance models can be visualized as graphs. Where provenance, dependency provenance can be visualized as bipartite graphs linking parts of the output with the parts of input. Why and how provenance graphs are more complex, but can be visualized as directed acyclic graph linking the parts of output to the input, where nodes are represented as algebraic operations. For workflow and database provenance the graphs provide a natural common formalism.

It is needed a common language that can express workflows and database queries. This paper introduces a graphical model of provenance for both database queries and simple workflows in a uniform way. This provides a foundation for studying complex workflow language features such as concurrency, while-loops and non-determinism.

II. BACKGROUND

The dataflow language DFL[13] is an extension of Nested Relational Calculus(NRC) that includes atomic functions and values. The more specification about the DFL and NRC has been found in [13,14]. The syntax of DFL is as follows:

$$\begin{aligned} e, e' \quad ::= \quad & x \mid \mathsf{let}\ x = e\ \mathsf{in}\ e' \mid c \mid f(e_1, \ldots, e_n) \\ \mid \quad & \pi_A(e) \mid \langle A_1{:}e_1, \ldots, A_n{:}e_n \rangle \mid \mathsf{empty?}(e) \\ \mid \quad & \mathsf{True} \mid \mathsf{False} \mid \mathsf{if}\ e\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 \\ \mid \quad & \emptyset \mid \{e\} \mid e_1 \cup e_2 \mid \{e' \mid x \in e\} \mid \bigcup e \end{aligned}$$

Here f denotes function and c denotes a constant atomic data value, drawn from a set D. Atomic data values may be values of base types such as integers or Boolean or strings, but they may also be more complicated objects such as data files or images. Functions include operations on basic data types, such as integer addition or equality. Furthermore, functions can also represent large computational steps such as external program or service calls: for example, to model the first provenance challenge workflow base types can be used such as Image, Header or WrapFile and function symbols such asalign_warp: Image*Header*Image*Header->WarpFile or reslice: WarpFile->Image*Header to represent the computation steps.

In the above syntax, the remaining syntactic constructs are standard components of the Nested Relational Calculus: It has been included Booleans,Conditionals and set

operations, record and field projection operations. The syntax $\{e'(x) \mid x \in e\}$ for the set comprehensions operation orfor-loop which evaluates e to a set $\{V_1,\ldots V_n\}$ and returns theset of values $\{e'(v_1),\ldots,e'(v_n)\}$ obtained by evaluating e' with x bound to each V_i. The expression $\bigcup e$ indicates a nested collection. The expression empty?(e) tests whether collection e is empty.

For the first and second projections of an ordered pair, it has been used ordered pair syntax $(e_1,e_2)$to abbreviate <fst:e1, snd:e2>, and write fst(e) or snd(e) instead of $\prod_{fst}(e)$ or $\prod_{snd}(e)$, respectively.

DFL and NRC are statically typed languages with an arbitrary but fixedsignature that assigns types to the constants and function symbols [14]and an arbitrary but fixed collection of atomic types. The static typing discipline ensures that expressions are always well-defined on input values of the correct type.

III.VALUE, EVALUATION AND PROVENANCE GRAPHS

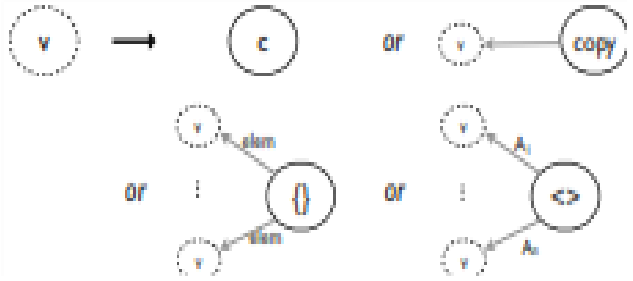
Evaluation of DFL expressions has been done over complex values, which are tuples of complex values $<A_1:V_1,\ldots A_n:V_n>$, nested combinations of atomic data values d and set of complex values $\{V_1,V_2,\ldots\ldots V_n\}$. As it has been shown in the next section A, it can easily be represented complex values as directed acyclic graphs or as trees. Using such value graphs, it has been represented the evaluation of a DFL expression by means of a provenance graph in the section C. A provenance graph is a two-sorted graph, consisting of a value graph and an evaluation graph, an evaluation graph has been introduced in the section B.

A. VALUE GRAPHS

A value graph is a directed acyclic graph(V,E) with labels on the nodes and edges. The nodes are represented using the alphabet $\{\{\}, \langle\rangle, \text{copy}\} \cup D$The edges are optionally represented using the alphabet $\{\text{elem}\} \cup \text{Attr}$. The formula $\text{lab}_l(n)$ to indicate that *n* has label *l* in G and $n \xrightarrow{l} n'$ to indicate that there is an edge (n, n') with label *l* in G.

To illustrate, the following graph represents the value $\{\langle A:1, B:2\rangle, \langle A:1, B:3\rangle\}$:

The legal value graph can be constructed using the following rules:



The meaning of these patterns is that a value graph v can be constructed from another valid graph by adding new nodes and edges (shown using solid lines) linked to some existing nodes (shown using dotted lines). The union of two disjoint value graphs is valid and also the empty graph is valid.Value tree is a tree-shaped value graph. Complex value can canonically be represented by the root node of a value tree. By duplicating shared nodes and by merging copy nodes with their targets any value graph can be converted to a value tree.

B. EVALUATIONGRAPHS

An evaluation graph G = (V; E) is a labelled directed acyclic graph with node labels selected from the set

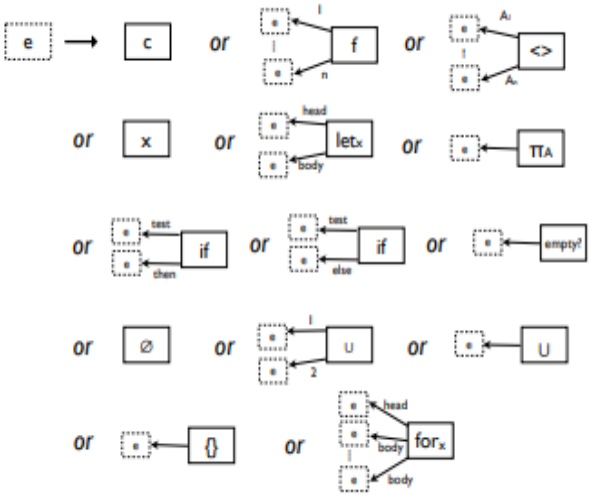
$$\{x, c, f, \langle\rangle, \pi_A, \text{let}_x, \text{if}, \emptyset, \{\}, \cup, \bigcup, \text{for}_x\}$$

And optional edge labels selected from the set

$$\{A, \text{head}, \text{body}, \text{test}, \text{then}, \text{else}, 1, 2, \ldots\}$$

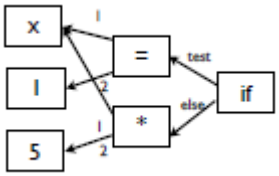
As with value graphs, $\text{lab}_l(m)$has writtento indicate that node *x* has label *l* and $m \xrightarrow{l} m'$ to indicate that nodes m and m' are linked by an edge represented by *l*.

Using the following rules a valid evaluation graph can be constructed.



The graph can be extended by adding new nodes and edges (shown using solid lines) by linking to existing nodes (shown using dotted lines). Sharing among the nodes of the evaluation graph is allowed. The union of two disjoint value graphs is valid and also the empty graph is valid.Finally, we introduce the following terminology: A node labelled let x or for x is said to bind x. It can be demonstrated that a variable node e_x labelled by x is in the scope of a node ethat binds x, if there is a path from e to e_x that does not pass through another node that binds x. It is required that each variable node to be in the scope of at most one binding node.

The following is a valid evaluation graph:

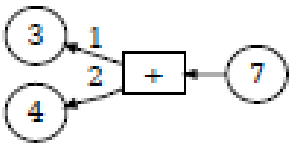


The above graph says that a value was obtained by doing a conditional test x = 1 which failed, and then evaluating the else-branch to return x* 5. Note that there is no information about the actual value of x or the result of the computation, although $x \neq 1$ holds.

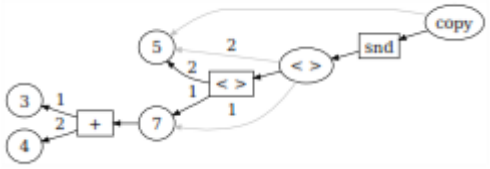
C. PROVENANCEGRAPHS

A provenance graph G = (V, E, *val*) is a directed acyclic graph with nodes and edges labelled with either value or evaluation labels, such that: 1.$V = V_V \uplus V_E$, 2. prov is a partial injective function from V_V to V_E so that each evaluation node e has a unique value node *val(e),* 3. G is value graph, If evaluation graph is disregarded,4. G is an evaluation graph if each pair of nodes (e, *val(e)*) is merged and disregard the value structure. In the following examples all the inputs are represented using gray boxes in the input expression.

The following example showing thecomputationof $3+4=7$:

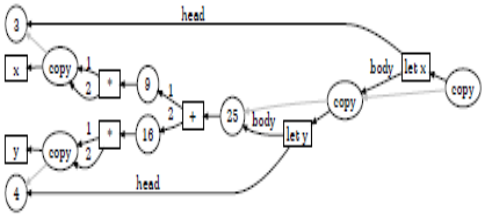


Here is a more complicated example, showing the evaluation of expression $\pi_2(3+4,5) = 5$involving constructing a pair and then selecting the second argument:



Finally, here is a larger example demonstrating let-binding, showing the evaluation of an expression

let x = 3 in let y = 4 in x* x + y* y.



IV. RELATED WORK

The given graphs are attempted to remain close to do visualizing provenance as graphs, particularly the Open Provenance Model[12], which distinguishes between process(evaluation) and artifact(value) nodes. The effort has not been done to make graphs fit OPM exactly. There are proposals for describing the provenance of collections in OPM is under development [15]. OPM is a representation format for provenance information.

Models of provenance have been studied in formal detail for some systems. Sroka et al[16] develop a semantics for Taverna workflows based on a core language. Missier et al[17] discusses lightweight lineage annotations for Taverna workflows. Graphical notations for provenance have been used extensively in many systems.

V. CONCLUSION

This paper has introduced the provenance for both database and workflow systems. It has been described in detail about the semantics that evaluates dataflow calculus expressions to provenance graphs containing values, evaluation nodes and links showing how the expression evaluated.

ACKNOWLEDGEMENT

I am grateful to my parents for their support and encouragement.

REFERENCES

[1] CUI, Y., WIDOM, J., AND WIENER, J. L. Tracing the lineage ofview data in a warehousing environment. ACM Trans. DatabaseSyst. 25, 2 (2000), 179–227.

[2] BUNEMAN, P., KHANNA, S., AND TAN, W. Why and where: Acharacterization of data provenance. In ICDT (2001), no. 1973 inLNCS, Springer, pp. 316–330.

[3] BUNEMAN, P., CHENEY, J., AND VANSUMMEREN, S. On theexpressiveness of implicit provenance in query and update languages. ACM Transactions on Database Systems 33, 4 (November2008), 28.

[4] BUNEMAN, P., KHANNA, S., AND TAN, W. On propagationof deletions and annotations through views. In PODS (2002),pp. 150–158.

[5] CHENEY, J., AHMED, A., AND ACAR, U. A. Provenance asdependency analysis. In DBPL 2007 (Vienna, Austria, September2007), M. Arenas and M. I. Schwartzbach, Eds., no. 4797 inLNCS, Springer-Verlag, pp. 139–153.

[6] CHENEY, J., ACAR, U. A., AND AHMED, A. Provenance traces.CoRR abs/0812.0564 (2008).

[7] GREEN, T. J., KARVOUNARAKIS, G., AND TANNEN, V. Provenances emirings. In PODS (2007), ACM, pp. 31–40.

[8] FOSTER, J. N., GREEN, T. J., AND TANNEN, V. AnnotatedXML: queries and provenance. In PODS (2008), pp. 271–280.

[9] ANAND, M. K., BOWERS, S., MCPHILLIPS, T., ANDASCHER, B. Efficient provenance storage over nested datacollections. In EDBT (New York, NY, USA, 2009), ACM,pp. 958–969. LUD

[10] SIMMHAN, Y. L., PLALE, B., AND GANNON, D. Karma2:Provenance management for data-driven workflows. Int. J. WebService Res. 5, 2 (2008), 1–22.

[11] SUN, P., LIU, Z., DAVIDSON, S. B., AND CHEN, Y. Detectingand resolving unsound workflow views for correct provenanceanalysis. In SIGMOD (New York, NY, USA, 2009), ACM,pp. 549–562.

[12] MOREAU, L., FREIRE, J., FUTRELLE, J., MCGRATH, R. E.,MYERS, J., AND PAULSON, P. The open provenance model: Anoverview. In Freire et al. [12], pp. 323–326.

[13] HIDDERS, J., KWASNIKOWSKA, N., SROKA, J.,TYSZKIEWICZ, J., AND VAN DEN BUSSCHE, J. DFL: Adataflow language based on petri nets and nested relationalcalculus. Inf. Syst. 33, 3 (2008), 261–284.

[14] BUNEMAN, P., NAQVI, S. A., TANNEN, V., AND WONG, L. Principles of programming with complex objects and collectiontypes.Theor. Comp. Sci. 149, 1 (1995), 3–48.

[15] GROTH, P., MILES, S., MISSIER, P., AND MOREAU, L. Aproposal for handling collections in the open provenance model,2009.

[16] SROKA, J., HIDDERS, J., MISSIER, P., AND GOBLE, C. Aformal semantics for the taverna 2 workflow model. Journal ofComputer and System Sciences In Press, Corrected Proof (2009).

[17] MISSIER, P., BELHAJJAME, K., ZHAO, J., ROOS, M., ANDGOBLE, C. A. Data lineage model for taverna workflows withlightweight annotation requirements. In Freire et al. [12], pp. 17–30.