Improving Workflow Design Using Abstract Provenance Graphs

Tianhong Song[1(✉)], Saumen Dey[1], Shawn Bowers[2], and Bertram Ludäscher[1]

[1] Department of Computer Science, University of California, Davis, Davis, USA
thsong@ucdavis.edu
[2] Department of Computer Science, Gonzaga University, Spokane, USA

1 Introduction and Motivation

A scientific workflow consists of a series of structured activities and computations that arise in scientific problem-solving. Recent work [7] has demonstrated that collection-oriented modelling and design (COMAD) [3] leads to simpler and more robust workflow design. In COMAD, for example, each actor is wrapped with a well defined configuration that hides the low level complexities of "wiring" processes together with respective data sources. On the other hand, some dataflow details (e.g., fine-grained data dependency information) are hidden in the workflow graph that the user may construct an erroneous or unoptimized workflow due to lack of information. Such problems are difficult to detect before workflow execution. Hence, configuring, maintaining and designing a collection-oriented workflow is sometimes challenging and time-consuming, in addition, large-scale workflows often tend to run for long time, therefore, error free and optimized workflow design is always desired.

Several approaches have been developed to detect and resolve workflow design issues in order to improve the correctness and efficiency of workflows, e.g., based on graph traversal [4], graph reduction [5], and graph refactoring [2]. Similar work has been done in the domain of Business Process Management [1] and Process-aware Information Systems [6]. However, much less work focuses on collection-oriented workflows. In previous work [8] *abstract provenance graphs* (APGs) have been proposed as a means to detect workflow design issues. APGs summarize all possible concrete provenance graphs and are computed prior to workflow execution via a static analysis technique, where fine-grained dependency information can be found.

Here, we extend this approach and illustrate how fine-grained data-oriented APGs can be used to improve workflow design by graph querying and pattern matching. Specifically, we propose to improve the state-of-the-art of workflow design and analysis and report on this ongoing work on both of the following fronts by employing abstract provenance information to: (1) Detect design problems. We show a declarative approach that workflow analysis can be performed by specifying and applying a set of queries and properties on APGs to detect design problems. (2) Improve workflow design. We propose to exploit APGs to derive a more parallel workflow structure prior to execution and to discover other optimization opportunities.

B. Ludäscher and B. Plale (Eds.): IPAW 2014, LNCS 8628, pp. 226–228, 2015.
DOI: 10.1007/978-3-319-16462-5_20

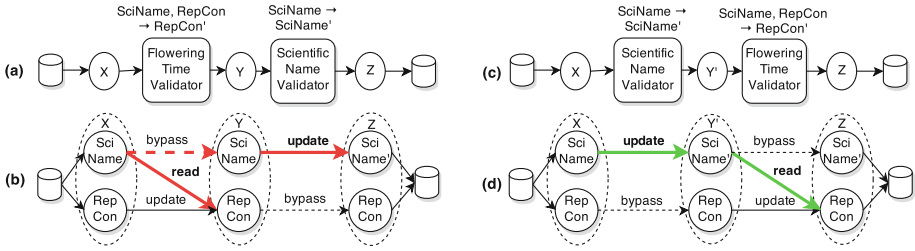

Fig. 1. Example workflow with actor configurations (a) and its inferred APG (b) showing fine-grained data dependencies. Highlighted edges reveal *read-before-updated (RBU)* pattern matching the query ($\mathsf{read}^{-1}.\mathsf{bypass}^{*}.\mathsf{update}$), indicating a design problem. Example workflow without the same problem (c) and its inferred APG (d).

We use data curation workflows as an example to showcase our work. A data curation workflow consists of several actors (processes) representing data validation tasks and connections among them indicating how the data flows. The data stream flowing among actors consists of a set of records (collections), each record contains a set of attribute-value pairs and each value is a concrete data item (e.g. scientific name). Each actor in a data curation workflow reads a data stream and validates and updates certain data items ("update") which may use other data items as references ("read") and the remaining (irrelevant) data items will be automatically transported bypassing the actor ("bypass").

2 Our Approach and Example

After an APG is constructed, design problems can be detected by checking whether the graph violates certain rules or constraints. Some of the problems can be recognized as graph patterns, e.g., certain type of edge cannot occur after another type. In this way, these problematic patterns can be discovered using graph queries. If the result of certain query is empty, this indicates a certain pattern is not present in the graph. So the corresponding workflow does not have this type of problem, otherwise, the workflow has this problem. Then we can inform the user about the design problems and suggest ways to fix them.

APGs can also be used to improve workflow design. Instead of constraints, optimization opportunities can be discovered by graph queries. If the result of a query is not empty, this indicates certain type of optimization opportunity has not been fully exploited. In addition, after the corresponding workflow has been improved, the same query can be applied again to check whether this type of opportunity has been fully exploited or not.

As the first prototype, we have identified some example constraints and optimization opportunities that can be discovered by Regular Path Query. Here, we show an example constraint called "No read-before-update". An example data curation workflow with two actors is shown in Fig. 1(a) where the "Scientific Name Validator" in the workflow validates "SciName" and the "Flowering Time

Validator" validates "RepCon" using "SciName" as a reference. This workflow has a problem that if "SciName" is not valid in the original input collection, then "Flowering Time Validator" may yield incorrect result due to the invalid reference. So sometimes we need to enforce a constraint that the data items used as references should be validated first. If we query the APG in Fig. 1(b) using this query $\mathsf{read}^{-1}.\mathsf{bypass}^{*}.\mathsf{update}$[1], the result will not be empty since "SciName" is read before it is validated. So we can conclude that the example workflow violates the "No read-before-update" constraint. The workflow shown in Fig. 1(c) doesn't have this issue (the order of actors is different), then if we apply the same query on the inferred APG in Fig. 1(d), the result is empty. Also, queries can be used to check whether a problematic workflow has been fixed or not.

Acknowledgements. Supported in part by NSF DBI-0960535 and ACI-0830944.

References

1. Brogi, A., Corfini, S., Popescu, R.: Semantics-based composition-oriented discovery of web services. ACM Trans. Internet Technol. (TOIT) **8**(4), 19:1–19:39 (2008)
2. Cohen-Boulakia, S., Chen, J., Missier, P., Goble, C., Williams, A.R., Froidevaux, C.: Distilling structure in Taverna scientific workflows: a refactoring approach. BMC Bioinform. **15**(Suppl. 1), S12 (2014)
3. McPhillips, T., Bowers, S., Zinn, D., Ludäscher, B.: Scientific workflow design for mere mortals. Future Gener. Comput. Syst. **25**(5), 541–551 (2009)
4. Meda, H.S., Sen, A.K., Bagchi, A.: On detecting data flow errors in workflows. J. Data Inf. Qual. (JDIQ) **2**(1), 4 (2010)
5. Sadiq, W., Orlowska, M.E.: Analyzing process models using graph reduction techniques. Inf. Syst. **25**(2), 117–134 (2000)
6. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features-enhancing flexibility in process-aware information systems. Data Knowl. Eng. **66**(3), 438–466 (2008)
7. Zinn, D.: Modeling and Optimization of Scientific Workflows. Ph.D. thesis, UC Davis, Davis, California (2010)
8. Zinn, D., Ludäscher, B.: Abstract provenance graphs: anticipating and exploiting schema-level data provenance. In: McGuinness, D.L., Michaelis, J.R., Moreau, L. (eds.) IPAW 2010. LNCS, vol. 6378, pp. 206–215. Springer, Heidelberg (2010)

[1] Reversed read edge followed by an update edge with zero or more bypass edges in-between.