Abstraction and Explanation in Collaborative Workflows

Serge Abiteboul
INRIA Saclay & ENS Cachan
serge.abiteboul@inria.fr

Pierre Bourhis
CNRS CRIStAL UMR 9189,
Lille
pierre.bourhis@univ-lille1.fr

Victor Vianu
U.C. San Diego
vianu@cs.ucsd.edu

1. INTRODUCTION

Process-centric workflows focus on control flow, often abstracting away data almost entirely. In contrast, recently proposed data-driven workflows treat data as first-class citizens, e.g., the *business artifact model* pioneered in [16] and deployed by IBM in commercial products. Data-driven workflows have become ubiquitous in a wide array of application domains. Their system architecture may range from totally centralized to fully distributed. While multiple-peer workflows have been extensively studied in the process-centric case using finite-state models, little formal research has been done on collaborative workflows centered around a database, which have infinitely many states (see related work).

In a previous paper [6], a model of *collaborative data-driven workflows* was introduced. In a local-as-view style, each peer has a partial view of a global instance that remains purely virtual. Local updates (insertions and deletions expressed as datalog-style rules) have side effects on other peers' data, defined via the global instance. The paper studies the ability of a peer to carry out runtime reasoning about the global run of the system, and in particular about actions of other peers, based on its own local observations. Serious restrictions are imposed to enable such reasoning.

In the present paper, we introduce a richer model for collaborative data-driven workflows based on the concept of "abstraction", and focus on the problem of explaning to a particular peer what is going on in the general system, in particular what the other peers are doing. From a data viewpoint, this is simple: the peer sees an abstraction of the global data at each instant in time. But we would like to also explain the events (that typically result in data updates) to the peer. At a global level, such an event is a rule instantiation. To explain the event, one can recursively explain how the positive and negative atoms in its body were obtained. (One can think of that as the *provenance* of the event.)

This provides an explanation of the event (that may not be unique). At the peer level, we will use an "abstraction of the explanation". This will ultimately allow us to build for a particular peer a customized workflow, that may be seen as the "abstraction of the global workflow".

2. ABSTRACTION

In a collaborative workflow environment, it is useful to explain to each peer the work of other peers. This can be trivially done by providing the full specification of the system. However, this is often undesirable, because (i) peers may not wish to fully share their data and specifications and (ii) peers may prefer not be overwhelmed with details about other peers. Instead, one can equip each peer with an *abstraction* that provides partial information of interest about the data that is shared and actions of the other peers.

A collaborative schema therefore specifies the abstraction that each peer sees. The *abstraction* of a relation for a given peer allows (i) hiding tuples (selection), (ii) hiding attributes (projection), and (iii) hiding data details (homomorphism). The third item is a particularity of the model. A homorphism (for a specific peer) hides certain information. For instance, it may replace the exact name of a product by its category, e.g. *electronic piano* and *smartphone* by *electronics*. Most interestingly in our context, it may hide the identity of a peer, so that a specific referee may simply become "referee".

We adopt an important restriction from [6]. Each relation has a key and an abstraction either hides entirely a relation or maintains the key. We can hide data details selectively on some columns, but never on the key. It is important to observe that (notably because of the presence of key), the peer does not perform view updates but direct updates to the global database. A standard *chase* is performed to take the key constraints into account [5], which may result in propagating the update to other peers because of the functional dependencies as well as the implicit inclusion dependencies between the global instance and the local views. (The implementation of such a propagation without recourse to a central authority with full access to the entire data is the topic of on-going work.)

3. EXPLANATION

In this section, we introduce the notion of explanation. This is a rather delicate process. Due to space limitations, many details are omitted.

Consider the trace $\tau = e_1, ..., e_n$ of actions from some start instance. We define the notion of *subtrace* as any subsequence that is actually a run. There are different ways of explaining how an operation e_i in τ was made possible. We consider primarily two: (i) find some minimal subtrace of τ

that ends with e_i; and (ii) find some minimal subtrace of τ that ends with e_i, "mimicking things that actually happened in τ" (to be explained further).

We study the problem for these two definitions, in particular, their semantics, algorithmic aspects and complexities.

For (ii), we will say that such a subtrace τ' that ends with e_i is "consistent with τ" if for each operation e in τ',

- (no-effect updates.) An update in the explanation has no effect, iff the corresponding update in the original run has no effect as well.
- (same cause literals) A literal used in some action in the explanation held continuously since some operation e' in τ' and not before, iff this was also the case for the corresponding literal in the original run.

With this definition, τ' explains what actually happened in the original trace in order to activate e_i.

Consider some peer P and a given explanation. The peer P sees an *abstraction* of the explanation, that is, a sequence of abstractions of operations. Thus, P sees only some of the data and portions of the rule instantiations from the actual workflow. In particular, some events may have no effect visible by P, but may eventually lead to some visible side effects. It should be observed that the resulting abstraction may not be the run of a real workflow, using valid rule instantiations: some rules may include logical contradictions. For instance, suppose that a rule says that if two managers from Human Resources disagree on hiring, the candidate is marked as "conflict". Such a rule instantiation at Human Resources and a possible abstraction seen by peer Bo may be:

```
@RH: marked@RH(Bo,conflict)
   :- forHiring@RH(Mary,Bo),
      not forHiring@RH(Paul,Bo)
@Bo: marked@RH(Bo,conflict)
   :- forHiring@RH(manager,Bo),
      not forHiring@RH(manager,Bo)
```

Here the abstraction is hiding the names of the managers who are handling Bo's hiring. Both names are mapped to the same value, namely "manager".

This may seen nonsensical logically. But one should consider it in the view of abstraction. For instance here, the term "manager" should be understood as a shorthand for some x such that $h(x, manager)$ where h is the homomorphism for *Bo*. So there is actually no contradiction.

We study the semantics of explanation. Ultimately, our goal is to be able to synthesize, given the global workflow and the abstraction of a particular peer, a workflow that explains the process as seen from that peer. Of course, we want this local workflow to be reasonably simple, to describe all possible runs (as seen from that peer), not to leak information, and to be as close as possible to what may actually happen.

4. RELATED WORK

Although not focused explicitly on workflows, Dedalus [7, 12] and Webdamlog [4, 2] are systems supporting distributed data processing based on condition/action rules. Local-as-view approaches are considered in a number of P2P data management systems, e.g., Piazza [17] that also consider richer mappings to specify views. Update propagation between views is considered in a number of systems, e.g., based on ECA rules in Hyperion [8].

Finite-state workflows with multiple peers have been formalized and extensively studied using communicating finite-state systems (called CFSMs in [1, 9], and *e-compositions* in the context of Web services, as surveyed in [13, 14]). Formal research on infinite-state, data-driven collaborative workflows is still in an early stage. The business artifact model [16] has pioneered data-driven workflows, but formal studies have focused on the single-user scenario. Compositions of data-driven web services are studied in [10], focusing on automatic verification. Active XML [3] provides distributed data-driven workflows manipulating XML data.

A collaborative system for distributed data sharing geared towards life sciences applications is provided by the Orchestra project [11, 15]. The underlying update propagation model among peers is based on schema mappings and is similar to our local-as-view approach.

5. REFERENCES

[1] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Inf. and Comp.*, 127(2), 1996.

[2] S. Abiteboul, E. Antoine, and J. Stoyanovich. Viewing the web as a distributed knowledge base. In *ICDE*, 2012.

[3] S. Abiteboul, O. Benjelloun, and T. Milo. The Active XML project: an overview. *VLDB J.*, 17(5), 2008.

[4] S. Abiteboul, M. Bienvenu, A. Galland, and E. Antoine. A rule-based language for web data management. In *PODS*, pages 293–304, 2011.

[5] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.

[6] S. Abiteboul and V. Vianu. Collaborative data-driven workflows: think global, act local. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*, pages 91–102, 2013.

[7] P. Alvaro, W. R. Marczak, N. Conway, J. M. Hellerstein, D. Maier, and R. Sears. Dedalus: Datalog in time and space. In *Datalog*, pages 262–281, 2010.

[8] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. J. Miller, and J. Mylopoulos. The hyperion project: from data integration to data coordination. *SIGMOD Record*, 32(3):53–58, 2003.

[9] D. Brand and P. Zafiropulo. On communicating finite-state machines. *JACM*, 30(2), 1983.

[10] A. Deutsch, L. Sui, V. Vianu, and D. Zhou. Verification of communicating data-driven web services. In *PODS*, 2006.

[11] T. J. Green, G. Karvounarakis, N. E. Taylor, O. Biton, Z. G. Ives, and V. Tannen. Orchestra: facilitating collaborative data sharing. In *SIGMOD*, 2007.

[12] J. M. Hellerstein. The declarative imperative: experiences and conjectures in distributed logic. *SIGMOD Record*, 39(1), 2010.

[13] R. Hull. Web services composition: A story of models, automata, and logics. In *ICSOC*, 2005.

[14] R. Hull and J. Su. Tools for composite web services: a short overview. *SIGMOD Record*, 34(2):86–95, 2005.

[15] Z. G. Ives, T. J. Green, G. Karvounarakis, N. E. Taylor, V. Tannen, P. P. Talukdar, M. Jacob, and F. Pereira. The orchestra collaborative data sharing system. *SIGMOD Record*, 37(3), 2008.

[16] A. Nigam and N. S. Caswell. Business artifacts: An approach to operational specification. *IBM Systems Journal*, 42(3):428–445, 2003.

[17] I. Tatarinov, Z. G. Ives, J. Madhavan, A. Y. Halevy, D. Suciu, N. N. Dalvi, X. Dong, Y. Kadiyska, G. Miklau, and P. Mork. The piazza peer data management project. *SIGMOD Record*, 32(3):47–52, 2003.