Provenance Composition in PROV

Provenance Composition in PROV

Peter Buneman[1], Adrià Gascón[1], Luc Moreau[2], and Dave Murray-Rust[1]

[1] School of Informatics, University of Edinburgh
[2] Electronics and Computer Science department, University of Southampton

Abstract. When two communicating processes each record their own provenance, what extra information needs to be recorded in order that a satisfactory account can be given, of the combined process? We propose a set of requirements on (i) the kind of information that the processes can share and (ii) the kind of queries that should be answerable from the combined provenance graph. We describe a solution using PROV.

1 Introduction

We examine the following scenario, which we believe may be quite common. Two processes, possibly long-running, communicate with one another. Each keeps an incremental record of its provenance. What, in addition to the two provenance stores do we need to keep in order to record the communication and to allow a satisfactory account of the provenance of combined processes? An example, given in [1], is of a research group communicating with a clinical data provider. Each side records its own provenance, but neither side wants to take responsibility for recording the other's provenance; nor – perhaps because of security or differing software environments – do they want to share their recording process with the other. However, they want to record enough about the communication that a provenance record of the entire process could be constructed if needed – perhaps because the research had discovered the existence of a patient at risk.

In a study of the problem of representing composition and substitution in graphical models of workflows and provenance [1], it was suggested that a graph model in which each node had *labelled ports* was needed in order to specify general operations such as composition and substitution on provenance graphs. What we propose here is a less general recipe based on message labelling for recording, in PROV, sufficient information about the communication that a satisfactory PROV account of the combined processes can be constructed.

While message passing has been studied in various proposals related to PROV and distributed provenance stores (a brief summary is given in Section 8), these proposals alone do not provide enough to address address the requirements we give here. In the closely related area of workflow models, abstractions have been proposed for the composition and substitution of workflows [2, 3]. Indeed, [3] investigates a model of workflows in which the purpose is to *conceal* the provenance associated with some part of the workflow in a series of executions of that workflow.

Our purpose in this paper is to provide a recipe for combining provenance descriptions formulated in PROV, and we assume a working familiarity with PROV. For the purpose of this explanation, it is enough to regard a PROV document as a set of *statements*, that correspond to the labeled edges in the its associated graph.

Our intention is to do this in a way that is transparent and would cause minimal effort for an organization that is already keeping a PROV record. That is, we want to make minimal assumptions about how PROV is used.

This paper is organised as follows. Section 2 defines five requirements that provenance composition in PROV needs to address. After defining some notations (Section 3), we propose a minimal solution, addressing two of these requirements. Then, Sections 5, 6 and 7 successively address the remaining requirements, building on PROV notions of bundle, attribution and ping-back, respectively. Related work, discussed in Section 8, is followed by the conclusions in Section 9.

2 Requirements

We want to be able to give a "satisfactory" account of the combined provenance, but what, in practice, does this mean? One could imagine some formal criteria based on reachability or causality [4], but this would not cover some of the practical issues such as responsibility for generating the relevant provenance data or for making sure that it is accessible. Instead we offer what we believe to be a practical set of requirements, with the understanding that some more formal analysis might be desirable. As mentioned above, in the scenario considered in this paper, we have two processes, or parties, A and B that communicate with each other.

1. The solution should use existing machinery – that belonging to A and B – for generating new URIs, and should not have to rely on a new "authority".
2. The parties should disclose as little information as possible about their local provenance, in particular URI naming schemes should be kept private.
3. When A sends a message m to B, B should be made aware of the location of A's provenance graph, even if B is denied access to that graph, i.e., a third party examining B's provenance should be able to find the provenance graph of the sender of m.
4. Information regarding origins should be preserved in the combined provenance graph, e.g. it should be easy to check whether a given entity in the joint provenance graph was generated by A or B.
5. The converse of requirement 3 should also hold: The provenance of A should be enough to identify the recipient of a message sent by A, and its associated provenance.

There are many solutions satisfying these general constraints. After introducing some notation, in Section 4 we describe what we believe is a minimal approach to this problem that satisfies requirements 1 and 2. Later, we extend our proposal to satisfy the rest of our requirements.

3 Notations

As above, consider two processes, or parties, A and B. If A sends some data D_1 to B, we assume that A denotes D_1 by an identifier, or URI, d_1^A; similarly, B denotes the received data D_1 as d_1^B. A subsequent piece of data D_2 from A to B will be denoted by URIs d_2^A and d_2^B and so on. Note that we use superscripts to denote the party that minted a given identifier. To be able to construct a satisfactory combined provenance graph, it is essential that we capture the pairing $(d_1^A, d_1^B), (d_2^A, d_2^B), \ldots$. This pairing, or correspondence, might be recoverable from accurate timestamping, but even if this is available, it will not be enough if messages can be sent in parallel. We shall use the term *combined provenance graph* to mean the result of taking union of the *local provenance graphs* of A and B, where nodes with the same identifier are merged.

4 A minimal approach to provenance composition in PROV

In this section, we focus on how to collect and represent, in PROV, the pairing $(d_1^A, d_1^B), (d_2^A, d_2^B), \ldots$ described in the previous section. Because we are using PROV to represent the provenance of A and B, the identifiers $d_1^A, d_1^B, d_2^A, d_2^B, \ldots$ are URIs generated by the two processes, or by whatever mechanism records the provenance of these two processes. As mentioned above, in general, we use superscripts to indicate the process that generated the URI. Thus, d_1^A is a URI generated by process A. From now on, when we refer to a piece of data d^A, or a message m^B, we mean, respectively, a PROV entity with URI d^A minted by A denoting some data D, and a PROV entity with URI m^B minted by B denoting a message.

Although we will explore this assumption in the next section, for the moment, we do not make any assumptions on how A (resp. B) records its interactions with B (resp A). Figure 1 illustrates the provenance graphs constructed by A and B, after data items D_1 and D_2 were communicated from A to B.

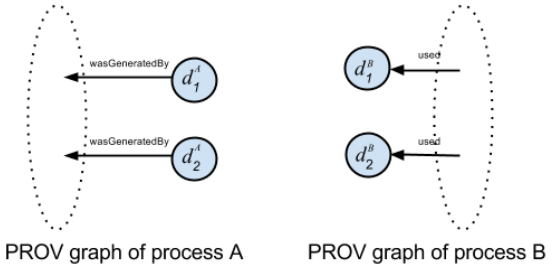

Fig. 1. The provenance graphs of processes A and B.

The entities denoted by d_1^A and d_1^B (resp. denoted by d_2^A and d_2^B) in Figure 1 may correspond to a variety of scenarios, such as a file being sent from A to B, or an acknowledgement being sent as part of a TCP data transfer from A to B. Even though d_1^A and d_1^B ultimately denote the same data, there are reasons for allowing them to have different URIs. For example a TCP packet on arrival may not be identical to what has been sent, or d_1^A may refer to some data that A subsequently modifies.

The main problem is that nothing in Figure 1 captures the pairing between d_1^A and d_1^B (resp. d_2^A and d_2^B). Thus, the fact that they respectively denote a data item that was communicated with a message between A and B is not captured in the graphs of Figure 1. We propose to represent this pairing in PROV by describing the messages as *entities* with URIs m_1^A and m_2^A in Figure 2. Moreover, we represent the fact that the message m_1^A contains the data d_1^A, also an entity, by means of the "wasDerivedFrom" relation. One could also represent represent other data relevant to the message, such as address of the recipient and destination port, with additional "wasDerivedFrom" PROV relations to the corresponding entities. Conversely, the data denoted by d_1^B is extracted from the message m_1^A, which is expressed by means of a further derivation again by means of an edge "wasDerivedFrom" from d_1^B to m_1^A, as shown in in Figure 2.

Note that the URIs of entities m_1^A, m_2^A, as denoted by their superscript, must be generated by process A. It is also important to note that both A and B use the same identifier to denote the message they exchanged. By simply taking the union of the PROV graphs in Figure 2, we get the combined PROV graph in Figure 3.

Note that requirement 2 is satisfied as long as the only URIs that are communicated between A and B are m_1^A and m_2^A. Indeed, the identifier d_1^A used internally by A

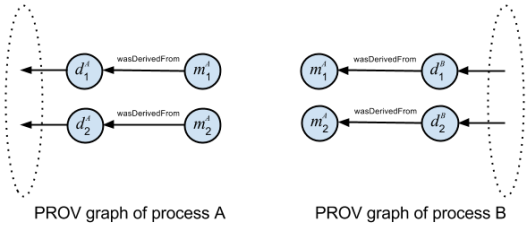

Fig. 2. Entities m_1^A and m_2^A denote the existence of messages sent from A to B.

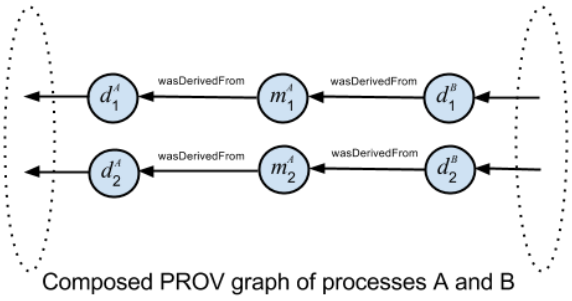

Fig. 3. The *combined* provenance graph of processes A and B in PROV.

to denote the data before communication does not have to be revealed to B, and vice versa.

Up to this point, we proposed that the URIs denoting messages, minted by the sender of the message, should be the only provenance-related information exchanged by the parties in addition to the data itself. The best way of exchanging such information depends on the specifics of the communication protocol. For that reason, we will not propose a concrete method for exchanging provenance information in this section. However, we will revisit this issue later on, when we address requirement 3.

5 Distributed provenance

Requirement 3, although simple to state intuitively, is not trivial to capture formally. Note that, when we require B's provenance to refer to A's provenance we are in fact refering to "provenance of provenance".

To this end, PROV introduces the concept of *bundle* [5], defined as a named set of provenance statements; it is a mechanism by which provenance of provenance can be expressed. With bundles, a provenance description can be given a name and can itself be regarded as an entity, whose provenance can in turn be described using PROV. Bundle names are URIs, which are also minted by the bundle creator, and can act as locator for the bundle, allowing its contents to be accessed according to some suitable access control. Section 6 deals specifically with the notion of attribution.

In a distributed setting, bundles, can be maintained at different locations by the various components of an application. In such a situation, bundles act as distributed islands of information that a provenance consumer needs to be able to navigate. For

instance, given a statement about m_1^A in a bundle produced by B, where can we find its provenance, which has been captured by A.

For this, we introduce the property `topicIn` [6] of an entity, which allows us to connect an entity in a given bundle, to another bundle where more provenance about that entity can be found.

Hence, a combination of bundles and the `topicIn` property are enough to address requirement 3. The solution requires A to create provenance descriptions related to the communication with B in a bundle[3], and share the bundle URI with B, like m^A was shared with B. Then, B can add a property `topicIn` to entity m^A, allowing navigation from B's bundle to A's bundle, referred to as *backward* navigation. In the following section, we show how to address requirement 4, and update our running example.

6 Attribution

For many purposes, a key consideration for deciding whether something is reliable and/or trustworthy is knowing who or what was reponsible for its production. Data published by a respected independent organization may be considered more trustworthy than that from a lobby organization; a claim by a well-known scientist with an established track record may be more believed than a claim by a new student; a calculation performed by an established software library may be more reliable than by a one-off program.

In our case, if the team performing analysis of anonymised clinical data identifies a patient at risk, it would need to find which organisation provided the clinical data about the patient.

PROV defines *attribution* as the ascribing of an entity to an agent [5]. Likewise, an *activity association* is an assignment of responsibility to an agent for an activity, indicating that the agent had a role in the activity [5]. Moreover, bundles can also be seen as entities, and therefore can be attributed to an agent. This gives us the necessary machinery to address requirement 4. Figure 4 shows an updated version of Figure 3, where the presented PROV machinery for fullfilling requirements 3 and 4 has been incorporated.

Adria: Labels in figures should be enlarged and placed properly.

7 Forward reachability and provenance pingback

To satisfy Requirement 3, A has to share the identifier of the provenance bundle it created, as well as B has to add the `topicIn` property to the description of message m_1^A it received from A, so that backward navigation from B's bundle to A's bundle becomes possible. Requirement 5 mandates the converse: the description of message m_1^A in A's bundle should also include a `topicIn` property pointing to the bundle of B containing further provenance description about the message m_1^A received from A. This in turn allows for *forward navigation* (or *forward reachability*)of B's bundle from A's bundle.

[3] The solution is flexible as it allows A to decide how to organise its provenance according to messages and recipients in bundles.

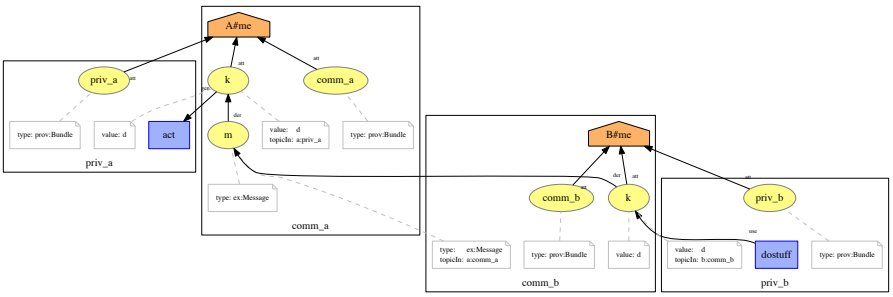

Fig. 4. Combined provenance graph with bundles and recorded attribution of a message sent from A to B. Party A contructs a bundle called `comm_a` containing descriptions of entities for the message being sent and its data. Both such bundle and the message's data are attributed to an agent representing A. Also, the provenance of the data of the message is associated to a bundle `priv_a`. This addresses requeriments 2 and 4. Analogously, B constructs a bundle called `comm_b` attributed to an agent representing B. Note that, in B's provenance, the message has a `topicIn` property refering to A's `comm_a` bundle, which allows B to trace the provenance of message m to A's provenance, fulfilling requirement 3. As all URIs are minted by either A or B, requirement 1 is also fulfilled.

For this to happen, A needs to be made aware of the bundle containing B's provenance descriptions. If the location of B's bundle was agreed before communication (e.g., by negotiation or at application design time), A can directly express the `topicIn` property as its provenance is being generated. In the more common circumstances in which the location was not agreed before communication, there is a requirement put on B to communicate the location of its bundle back to A: this requirement necessitates a message to be passed from B to A.

Two cases are then possible: either B's bundle location can be piggybacked onto a subsequent application message from B to A, or an extra message solely for the purpose of sharing this information can be sent. While the former approach does not require extra messages to be exchanged, forward reachability guarantees are difficult to be made, since it is dependent on the application's behaviour to pass that information (B may be very slow to respond back to A, for instance). In the latter option, dedicated messaging allows for prompt availability of forward reachability. PROV-AQ [7] offers a mechanism to support this functionality referred to as *pingback*: when sending m_1^A, A can share the address of a pingback endpoint, to which B can post the location of its bundle. Upon receiving a bundle location, A's pingback endpoint can enrich A's provenance descriptions with a `topicIn` property.

8 Related provenance models

Workflow systems such as Taverna [8] provide facilities for the composition of workflows and for the representation of subroutines. Also, in [9] a method is described for labelling that allows modules to be recursively expanded (a form of graph substitution). In these models, the basic module is described with labelled inputs and outputs. In this paper, we have used message identifiers as these labels. Using global identifiers as labels has some limitations – it does easily permit any kind of graph substitution or recursive expansion – but it is sufficient for what we want to do here, which is to compose existing provenance graphs. A more general solution based on

port labelling [1] could be simulated in PROV, but it would substantially complicate the solution.

Message identifiers are also used in the Provenance-Aware Service Oriented Architecture (PASOA [10, 11]), in which so-called *event identifiers* allow the linking of assertions independently made by the senders and receivers of messages, in a structure called the *p-structure*. Such p-structures can be recorded in distributed provenance stores. PASOA defines a notion of *p-header* [11] as a header to be inserted in messages and containing provenance-specific information: a p-header includes a message identifier and a list of URIs denoting the provenance stores the sender used to record provenance in. Such URIs allow the distributed p-structure to be navigated both backward and forward.

While PASOA [10, 11] was framed in the context of service-oriented architectures, and specifically of Web Services, the Open Provenance Model (OPM [12]) adopted a conceptual data model to define provenance. In this context, the mechanisms defined by PASOA [10, 11] were re-framed in OPM [13]: on the sending side, a message to be sent (described as an entity in PROV terminology [5]) is derived from the data inserted in it; on the receiving side, data received is extracted from a received message. The link between a received message and a sent message is established by its identifier. This approach [13] shares several aims with this paper, but pre-dates the Recommendation PROV [5], and therefore is not standardised. Furthermore, it fails to deconstruct the solution according to the requirements it seeks to address.

PROV [5] introduces two notions absent from OPM used in this paper. First, PROV has a notion of attribution, allowing an entity to be ascribed to an agent; second, PROV introduces the concept of *bundle* [5], which we used in our proposal to ensure that provenance can be traced backwards across systems boundaries. To this purpose, we also use the `topicIn` relation introduced by Moreau and Groth [6]. This relation is used to to associate an entity with a bundle, in which further descriptions about that entity can be found. The relation `topicIn` bears some similarity with `mentionOf` [14], except that it is not defined as a subproperty of specializationOf [5] (which therefore was irreflexive, and did not allow us to refer to the same entity occurring in a separate bundle).

While PASOA's p-header introduced the idea of embedding provenance information in messages, PROV-AQ [7] provides a partial instantiation of this idea in the context of HTTP. For a resource accessible using HTTP, the location of its provenance may be indicated by using the HTTP Link header field [15] `has_provenance`, referring to a URL where further provenance can be found. Note that PROV-AQ [7] only allows such Link header field to be included in the HTTP response to a GET or HEAD operation, and it is unclear whether this approach would be sufficient for the solution we propose here.

PROV-AQ also introduces a "pingback" mechanism by which any client is allowed to report the existence of a provenance about a given resource [7]. The "pingback" mechanism is declared by another HTTP Link header field, `pingback`, providing a URL to which location of provenance can be posted to. WebMention [16] is a normative mechanism enabling similar posting back of URLs to allow forward navigation of documents on the Web.

The approach we describe here keep the provenance separate from the data being communicated. Alternative approaches, such as Souilah et al. [17], enrich data with a provenance field that accumulates the send/receive operations it underwent. Given that provenance information is carried along with data, sharing of provenance

information is not possible for data products that have a common provenance. In contrast, graphical models such as PROV and OPM allow for such a sharing.

9 Concluding Remarks

We have constructed a solution according to five requirements that need to be addressed to allow provenance composition in PROV. First, we have identified a minimal solution by which a message identifier is shared by its sender and receiver, to allow a form of composition addressing the first two requirements. The minimal solution was then progressively enriched with bundles and the `topicIn` property, attribution and the pingback mechanism to address subsequent requirements successfully.

While the solution that we offer here addresses a gap in the PROV specifications, it still leaves open a number of practical issues. To ensure inter-operable behaviour, issues that need to be worked out in detail include the conventions to share identifiers, the normative definition of `topicIn`, and the ability to access (or dereference) bundles from their identifiers. From a foundational viewpoint, a more formal study of the protocol might be desirable in conjunction with the notion of labelled ports [1].

References

1. Buneman, P., Gascón, A., Murray-Rust, D.: Composition and Substitution in Provenance and Workflows. In: Proceedings of the 8th USENIX Worshop on the Theory and Practice of Provenance, TAPP 2016, June 8-9, 2016, Washington D.C. USA. (2016)
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 bioinformatics **15**(1) (2014) 1
3. Davidson, S.B., Khanna, S., Milo, T., Panigrahi, D., Roy, S.: Provenance views for module privacy. In: Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece. (2011) 175–186
4. Cheney, J.: Causality and the semantics of provenance. arXiv preprint arXiv:1004.3241 (2010)
5. Moreau, L., Missier (eds.), P., Belhajjame, K., B'Far, R., Cheney, J., Coppens, S., Cresswell, S., Gil, Y., Groth, P., Klyne, G., Lebo, T., McCusker, J., Miles, S., Myers, J., Sahoo, S., Tilmes, C.: PROV-DM: The PROV Data Model. W3C Recommendation REC-prov-dm-20130430, World Wide Web Consortium (April 2013)
6. Moreau, L., Groth, P.: Provenance: An Introduction to PROV. Morgan and Claypool (September 2013)
7. Klyne, G., Groth (eds.), P., Moreau, L., Hartig, O., Simmhan, Y., Myers, J., Lebo, T., Belhajjame, K., Miles, S.: PROV-AQ: Provenance Access and Query. W3C Working Group Note NOTE-prov-aq-20130430, World Wide Web Consortium (April 2013)
8. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-science: An overview of workflow system features and capabilities. Future Generation Computer Systems **25**(5) (2009) 528–540
9. Bao, Z., Davidson, S.B., Milo, T.: Labeling workflow views with fine-grained dependencies. Proceedings of the VLDB Endowment **5**(11) (2012) 1208–1219
10. Groth, P., Miles, S., Moreau, L.: A Model of Process Documentation to Determine Provenance in Mash-ups. Transactions on Internet Technology (TOIT) **9**(1) (2009) 1–31
11. Groth, P., Jiang, S., Miles, S., Munroe, S., Tan, V., Tsasakou, S., Moreau, L.: An architecture for provenance systems. Technical report, University of Southampton (February 2006)

12. Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E., Van den Bussche, J.: The open provenance model core specification (v1.1). Future Generation Computer Systems **27**(6) (June 2011) 743–756
13. Groth, P., Moreau, L.: Representing distributed systems using opm. Future Generation Computer Systems **27**(6) (June 2011) 757–765
14. Moreau, L., Lebo, T.: Linking across provenance bundles. W3C Working Group Note NOTE-prov-sem-20130430, World Wide Web Consortium (April 2013)
15. Nottingham, M.: Web linking. Internet RFC 5988, IETF (October 2010)
16. Parecki, A.: Webmention. W3C Recommendation REC-webmention-20170112, World Wide Web Consortium (January 2017)
17. Souilah, I., Francalanza, A., Sassone, V.: A formal model of provenance in distributed systems. In Cheney, J., ed.: TAPP'09: First workshop on on Theory and practice of provenance, San Francisco, CA, USENIX Association (February 2009)