The Open Provenance Model Core Specification (v1.1)

Luc Moreau (Editor)[a,*], Ben Clifford[b], Juliana Freire[c], Joe Futrelle[d], Yolanda Gil[e], Paul Groth[f], Natalia Kwasnikowska[g], Simon Miles[h], Paolo Missier[i], Jim Myers[d], Beth Plale[j], Yogesh Simmhan[k], Eric Stephan[l], Jan Van den Bussche[g]

[a] *U. of Southampton*
[b] *No Affiliation*
[c] *U. Utah*
[d] *NCSA*
[e] *Information Sciences Institute, USC*
[f] *VU University of Amsterdam*
[g] *U. Hasselt and transnational U. Limburg*
[h] *King's College, London*
[i] *U. of Manchester*
[j] *Indiana University*
[k] *Microsoft*
[l] *Pacific Northwest National Laboratory*

Abstract

The Open Provenance Model is a model of provenance that is designed to meet the following requirements: (1) To allow provenance information to be exchanged between systems, by means of a compatibility layer based on a shared provenance model. (2) To allow developers to build and share tools that operate on such a provenance model. (3) To define provenance in a precise, technology-agnostic manner. (4) To support a digital representation of provenance for any "thing", whether produced by computer systems or not. (5) To allow multiple levels of description to coexist. (6) To define a core set of rules that identify the valid inferences that can be made on provenance representation. This document contains the specification of the Open Provenance Model (v1.1) resulting from a community effort to achieve inter-operability in the Provenance Challenge series.

Keywords: provenance, representation, inter-operability

1. Introduction

Provenance is well understood in the context of art or digital libraries, where it respectively refers to the documented history of an art object, or the documentation of processes in a digital object's life cycle [1]. Interest for provenance in the "e-science community" [2] is also growing, since provenance is perceived as a crucial component of workflow systems [3] that can help scientists ensure reproducibility of their scientific analyses and processes.

Against this background, the *International Provenance and Annotation Workshop* (IPAW'06), held in Chicago, in May 2006, involved some 50 participants interested in the issues of data provenance, process documentation, data derivation, and data annotation [4, 5]. During a session on provenance standardization, a consensus began to emerge, whereby the provenance research community needed to understand better the capabilities of the different systems, the representations they used for provenance, their similarities, their differences, and the rationale that motivated their designs.

Hence, the first Provenance Challenge was born, and from the outset, the challenge was set up to be *informative* rather than *competitive*. The first Provenance Challenge aimed to provide a forum for the community to understand the capabilities of different provenance systems and the expressiveness of their provenance representations. Participants simulated or ran a Functional Magnetic Resonance Imaging workflow, from which they implemented and executed a pre-identified set of "provenance queries". Sixteen teams responded to the challenge, and reported their experience in a journal special issue [6].

The first Provenance Challenge was followed by the second Provenance Challenge, aiming at establishing interoperability of systems, by exchanging provenance information. Thirteen teams [7] responded to this second challenge. Discussions indicated that there was substantial agreement on a core representation of provenance. As a result, following a workshop in Salt Lake City, in August 2007, a data model was crafted and released as the *Open Provenance Model* (v1.00) [8].

*Corresponding author

The starting point of this work is the community agreement summarized by Miles [9]. We assume that provenance of objects (whether digital or not) is represented by an annotated causality graph, which is a directed acyclic graph, enriched with annotations capturing further information pertaining to execution. For the purpose of this paper, a provenance graph is defined to be *a record of a past execution* (or current execution), and not a description of something that could happen in the future.

In June 2008, twenty participants attended the first OPM workshop [10] to discuss the OPM specification v1.00. Minutes of the workshop and recommendations [11] were published, and led to version v1.01 of the Open Provenance Model [12], which was actively used during the Third Provenance Challenge, which aimed at exchanging provenance information encoded in OPM and answering precise provenance queries. Some 15 teams participated in this third challenge, and decided to adopt an open-source model for the governance[13] of OPM. A series of proposals were put forward, publicly reviewed, and put to vote [14]; the result of which is version 1.1 of the Open Provenance Model, which we present in this paper.

2. Requirements

The *Open Provenance Model* (OPM) is a model of provenance that is designed to meet the following requirements:

- To allow provenance information to be exchanged between systems, by means of a compatibility layer based on a shared provenance model.
- To allow developers to build and share tools that operate on such provenance model.
- To define provenance in a precise, technology-agnostic manner.
- To support a digital representation of provenance for any "thing", whether produced by computer systems or not.
- To allow multiple levels of description to coexist.
- To define a core set of rules that identify the valid inferences that can be made on provenance representation.

While specifying this model, we also have some *non*-requirements:

- It is not the purpose of this document to specify the internal representations that systems have to adopt to store and manipulate provenance internally; systems remain free to choose internal representations that are fit for their purpose.
- It is not the purpose of this document to define a computer-parsable syntax for this model; realisations of OPM in XML, RDF or others are being specified in separate documents.
- We do not specify protocols to store such provenance information in provenance repositories.
- We do not specify protocols to query provenance repositories.

3. Basics

The Open Provenance Model allows us to characterize what caused "things" to be, i.e., how "things" depended on others and resulted in specific states. In essence, it consists of a directed graph expressing such dependencies. We introduce here the constituents of such a graph.

3.1. Nodes

Our primary concern is to be able to represent how "things", whether digital data such as simulation results, physical objects such as cars, or immaterial entities such as decisions, came out to be in a given state, with a given set of characteristics, at a given moment. It is recognized that many of such "things" can be stateful: a car may be at various locations, it can contain different passengers, and it can have a tank full or empty; likewise, a file can contain different data at different moments of its existence. Hence, from the perspective of provenance, we introduce the concept of an *artifact* as an immutable[1] piece of state; likewise, we introduce the concept of a *process* as actions resulting in new artifacts.

A process usually takes place in some context, which enables or facilitates its execution: examples of such contexts are varied and include a place where the process executes, an individual controlling the process, or an institution sponsoring the process. These entities are being referred to as *Agents*. Agents, as we shall see when we discuss causality dependencies, are a cause (similar to a catalyst) of a process taking place.

The Open Provenance Model is based on these three kinds of nodes, which we now define.

Definition 1 (Artifact). *Immutable piece of state, which may have a physical embodiment in a physical object, or a digital representation in a computer system.*

Definition 2 (Process). *Action or series of actions performed on or caused by artifacts, and resulting in new artifacts.*

[1] In the presence of streams, we consider an artifact to be a slice of stream in time, i.e. the stream content at a specific instant in the computation. A future version of OPM will refine the model to accommodate streams fully as they are recognized to be crucial in many applications.

Definition 3 (Agent). *Contextual entity acting as a catalyst of a process, enabling, facilitating, controlling, or affecting its execution.*

The Open Provenance Model is a model of artifacts *in the past*, explaining how they *were* derived. Likewise, processes also occurred in the past, i.e. they have already completed their execution; in addition, processes can still be currently running (i.e., they may have not completed their execution yet). In no case is OPM intended to describe the state of future artifacts and the activities of future processes.

To facilitate understanding and promote a shared visual representation, we introduce a graphical notation for provenance graphs. Specifically, artifacts are represented by ellipses; processes are represented graphically by rectangles; finally, agents are represented by octagons.

3.2. Dependencies

The Open Provenance Model aims to capture the causal dependencies between the artifacts, processes, and agents. Therefore, a provenance graph is defined as a directed graph, whose nodes are artifacts, processes and agents, and whose edges belong to one of the following categories depicted in Figure 1. An edge represents a causal dependency, between its source, denoting the effect, and its destination, denoting the cause.

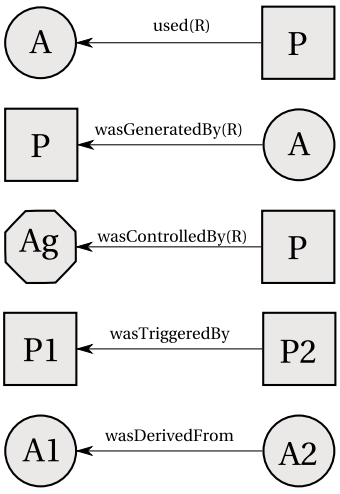


Figure 1: Edges in the Open Provenance Model: sources are effects, and destinations causes

The first two edges express that a process *used* an artifact and that an artifact *was generated by* a process. Since a process may have used several artifacts, it is important to identify the *roles* under which these artifacts were used. (Roles are denoted by letter 'R' in Figure 1.) Likewise, a process may have generated many artifacts, and each would have a specific *role*. For instance, the division process uses two numbers, with roles dividend and divisor, and produces two numbers, with roles quotient and rest. Hence, roles are similar to parameters of a function, except that they are used to distinguish inputs and outputs. Consequently, roles are meaningful only in the context of the process where they are defined. The meaning of roles is not defined by OPM but by application domains; OPM only uses roles syntactically (as "tags") to distinguish the involvement of artifacts in processes.

A process is caused by an agent, essentially acting as a catalyst or controller: this causal dependency is expressed by the *was controlled by* edge. Given that a process may have been controlled by several agents, we also identify their roles as controllers. We note that the dependency between an agent and a process represents a control relationship, and not a data derivation relationship. It is introduced in the model to express easily how a user (or institution) controlled a process.

Even though an artifact A_2 may have been generated by a process that used some artifacts, this does not tell us which artifact A_2 actually depends upon. Hence, to make this dependency explicit, it is required to assert that artifact A_2 was *derived from* another artifact A_1. This edge gives us a dataflow oriented view of provenance.

It is also recognized that we may not be aware of the exact artifact that a process P_2 used, but that there was some artifact generated by another process P_1. Process P_2 is then said to have been *triggered by* P_1. In contrast to edge *was derived from*, a *was triggered by* edge allows for a process oriented view of past executions to be adopted. (Since these edges summarize some activities for which all details are not being exposed, it was felt that it was not necessary to associate a role with them.)

As far as conventions are concerned, we note that causality edges use past tense to indicate that they refer to past execution. Causal relationships are defined as follows.

Definition 4 (Causal Relationship). *A causal relationship is represented by an arc and denotes the presence of a causal dependency between the source of the arc (the effect) and the destination of the arc (the cause).*

Five causal relationships are recognized: a process used an artifact, an artifact was generated by a process, a process was triggered by a process, an artifact was derived from an artifact, and a process was controlled by an agent. By means of annotations (see Section 8), we allow edges to be further subtyped from these five categories.

Multiple notions of causal dependencies were considered for OPM. A very strong notion of causal dependency would express that a set of entities was necessary and sufficient to explain the existence of another entity. It was felt that such a notion was not practical, since, with an open world assumption, one could always argue that additional factors may have influenced an outcome (e.g. electricity was used, temperature range allowed computer to work, etc). It was felt that weaker notions, only expressing *necessary dependencies*, were more appropriate. However, even then, one can distinguish data dependencies (e.g. where

a quotient is clearly dependent on the dividend and divisor) from a control dependency where the mere presence of some artifact or the beginning of a process can explain the presence of another entity. A number of factors have influenced us to adopt a weak notion of causal dependency for OPM.

- *Expressibility.* It is anticipated that systems will produce descriptions of what their components are doing, without having intimate knowledge of the exact internal data and control dependencies. Weak notions of dependency are necessary for such systems to be able to use OPM in practice.
- *Composability.* We shall see how OPM supports multi-level descriptions (Section 4). In a system consisting of the parallel composition of two subcomponents, the high-level summary of the system requires a weaker notion of dependency than the low-level descriptions of its subcomponents.

Hence, we adopt the following causal dependencies in OPM. We anticipate that subclasses of these dependencies, capturing stronger notions of causality, may be defined in specific systems, and over time, may be incorporated in OPM.

Definition 5 (Artifact Used by a Process). *A "used" edge from process to an artifact is a causal relationship intended to indicate that the process required the availability of the artifact to be able to complete its execution. When several artifacts are connected to a same process by multiple "used" edges, all of them were required for the process to complete.*

Alternatively, a stronger interpretation of the *used* edge could have required the artifact to be available for the process to be able to start. (Such an interpretation corresponds to a call-by-value procedure invocation where the arguments are required for the procedure to be invoked.) It is believed that such a notion may be useful in some circumstances, and it may be defined as a subtype of *used*. We note that both interpretations of *used* coincide, when processes are modelled as instantaneous. However, such a stronger notion is not compositional: an artifact A may have been required to begin execution of P_1, but it does not mean that A was required to begin P_2, a super-process of P_1.

Definition 6 (Artifacts Generated by Processes). *A "was generated by" edge from an artifact to a process is a causal relationship intended to mean that the process was required to initiate its execution for the artifact to have been generated. When several artifacts are connected to a same process by multiple "was generated by" edges, the process had to have begun, for all of them to be generated.*

A stronger interpretation is that the process had to complete for the artifact to be generated. This alternative interpretation was rejected because it made it difficult to model pipelined processes exchanging artifacts.

Definition 7 (Process Triggered by Process). *An edge "was triggered by" from a process P_2 to a process P_1 is a causal dependency that indicates that the start of process P_1 was required for P_2 to be able to complete.*

We note that the relationship P_2 *was triggered by* P_1 (like the other causality relationships we describe in this section) only expresses a *necessary* condition: P_1 was required to have started for P_2 to be able to complete. This interpretation is weaker than the common sense definition of "trigger", which tends to express a sufficient condition for an event to take place.

Definition 8 (Artifact Derived from Artifact). *An edge "was derived from" from artifact A_2 to artifact A_1 is a causal relationship that indicates that artifact A_1 needs to have been generated for A_2 to be generated. The piece of state associated with A_2 is dependent on the presence of A_1 or on the piece of state associated with A_1.*

Definition 9 (Process Controlled by Agent). *An edge "was controlled by" from a process P to an agent Ag is a causal dependency that indicates that the start and end of process P was controlled by agent Ag.*

3.3. Roles

Roles are constituents of "used", "was generated by", and "was controlled by" edges, aimed at distinguishing the nature of the dependency when multiple such edges are connected to a same process.

Definition 10 (Role). *A role designates an artifact's or agent's function in a process.*

A role is used to differentiate among several use, generation, or controlling relations.

1. A process may use (resp, generate) more than one artifact. Each "used" (resp, "was generated by") relation may be distinguished by a role with respect to that process. For example, a process may use several files, reading parameters from one (role = "parameters"), and reading data from another (role = "data").
2. An artifact might be used by more than one process, possibly for different purposes. In this case, the "used" relations can be distinguished by their associated roles. For example, a dictionary might be used by one process to look up the spelling of "provenance", (role = "look up provenance"), while another process uses the same dictionary to hold open the door (role = "doorstop").
3. An agent may control more than one process. In this case, the different processes may be distinguished by the role associated with the "was controlled by" relation. For example, a gardener may control the digging process (role = "dig the bed"), as well as planting a rose bush (role = "plant") and watering the bush (role = "irrigating").

4. A process may be controlled by more than one agent. In this case, each agent might have a distinct controlling function, which would be distinguished by roles associated with the "was controlled by" relations. For example, boarding the train may be controlled by the ticket agent (role = "sell ticket"), the gate agent (role = "take ticket") and the steward (role = "guide to seat").

From an OPM's perspective, roles have a syntactic nature and are scoped by the process which they are related to. A role has meaning only within the context of a given process (and/or agent). For a given process, each "used", "was generated by" or "was controlled by" relation has a role specific to the process, though the roles may have no meaning outside that process. OPM does not mandate the uniqueness of roles for a given process. For example, baking a cake with two eggs, may define each egg as a separate artifact, and the two used edges might have the identical role, say, egg. (In such a case, there is nothing that distinguishes the involvement of one egg from the other in this process.)

Roles should always be specified. For inter-operability, communities should define standard sets of roles with agreed meanings (by means of profiles, defined in Section 9). In addition, a reserved value is defined for "undefined", which should be used when the role is not known or omitted.

3.4. Examples

An example illustrating all the concepts and a few of the causal dependencies is displayed in Figure 2. The context of Figure 2 is the first Provenance Challenge [6], where an fMRI workflow operated on a series of images and headers, and produced an average image according to different axes. Figure 2 displays a subset of the provenance for one of the outputs "Atlas X Graphic", which was generated by an execution of First Provenance Challenge workflow using several inputs; the User who controlled this process was John Doe. Edges of type "used", "was generated by", and "was controlled by" are represented by dotted lines, annotated with their role in bracket. Data derivations are explicitly represented by "was derived from" edges, represented by plain lines. We note that the fact that a process used an artifact and generated another does not imply the latter was derived from the former; such relationship needs to be asserted explicitly.

OPM is in no way limited to digital artifacts and processes. In Figure 3, a provenance graph expresses that John baked a cake with ingredients butter, eggs, sugar and flour.

While graphs can be constructed by incrementally connecting artifacts, processes, and agents with individual edges, the meaning of the causality relations can be understood in the context of all the *used* (or *wasGeneratedBy*) edges, for each process. By connecting a process to several artifacts by *used* edges, we are not just stating the individual inputs to the process. We are asserting a causal dependency expressing that the process could take place and complete only because *all* these artifacts were available. Likewise, when we express that several artifacts were generated by a process, we mean that these artifacts would not have existed if the process had not begun its execution; furthermore, *all* of them were generated by the process; one could not have been generated without the others. The implication is that any single generated artifact is caused by the process, which itself is caused by the presence of all the artifacts it used. We will investigate transitive closures of causality relations in Section 6.

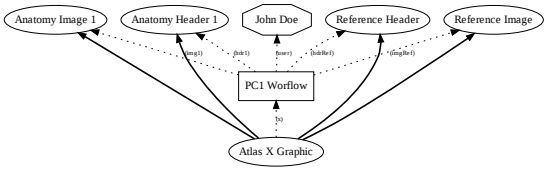


Figure 2: Provenance of "Atlas X Graphic" in First Provenance Challenge Workflow

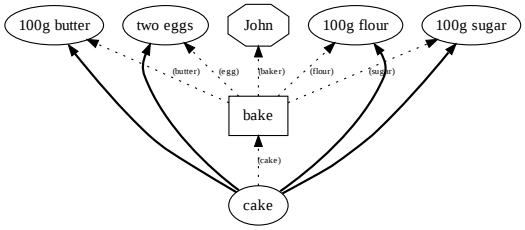


Figure 3: Victoria Sponge Cake Provenance

We can see here the crucial difference between artifacts and the data they represent. For instance, the *data* may have existed, but the particular *artifact* did not. For example, a BLAST search can be given a DNA sequence and return a set of "similar" DNA sequences; however, these returned sequences all existed prior to the process (BLAST) invocation, but the artifacts are novel.

As illustrated by the two examples above, the entities and edges introduced in Figure 1 allow us to capture many of the use cases we have come across in the provenance literature. However, they do not allow us to provide descriptions at multiple level of abstractions, or from different view points. To support these, we allow multiple descriptions of a same execution to coexist.

4. Overlapping and Hierarchichal Descriptions

Figure 4 shows two examples of provenance graphs describing what led the list (3,7) to being as it is. According to the left-hand graph, the list was generated by a pro-

cess that added one to all constituents of the list (2,6). According to the right-hand graph, the derivation process of (3,7) required the list to be created from values 3 and 7, respectively obtained by adding one to 2 and 6, themselves being the data products obtained by accessing the contents of the original list (2,6). To facilitate the understanding of these figures, edges of the type "was derived from" are subtyped, and their subtype made explicit as a label to the edge. (We will come back to the notion of subtyping in Section 8.)

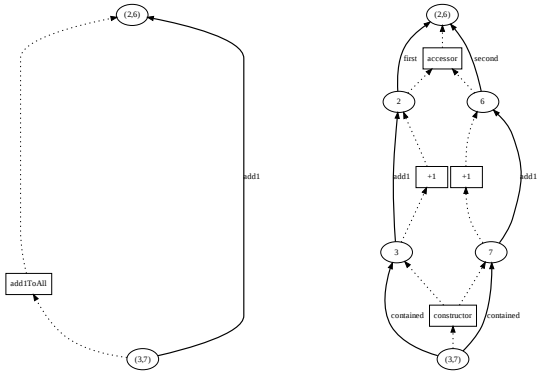


Figure 4: Examples of Provenance Graph

Assuming these two graphs refer to the same lists (2,6) and (3,7), they provide two different explanations of how (3,7) was derived from (2,6): these explanations would offer different levels of details about the same derivation. The requirement of providing details at different levels of abstraction or from different viewpoints is common for provenance systems, and hence, we would expect both accounts to be integrated in a single graph. In Figure 5, we see how the two provenance graphs of Figure 4 were integrated, by selecting different colors for nodes and edges. The lighter (red) part belonged to the left graph of Figure 4, whereas the darker (black) part is the alternate description from the right graph of Figure 4. (Graphs in this paper are better viewed in color.) The darker and lighter subgraphs are two different overlapping *accounts* of the same past execution, offering different levels of explanation for such execution. Such subgraphs are said to be *overlapping accounts* because they share some common nodes (2,6) and (3,7). Furthermore, the darker part (black) provides more details than the lighter subgraph (red): the darker part is said to be a *refinement* of the lighter graph. (The term 'refinement' is to be understood as a more complete description of execution, and is inspired by the concept of specification refinement in formal methods [15].)

Observing Figure 5, it becomes crucial to contrast the edges "was generated by" originating from artifact (3,7) with the edges "used" originating from the constructor process. Indeed, the edges "used" out of the constructor process mean that *both* artifacts 3 and 7 were required for

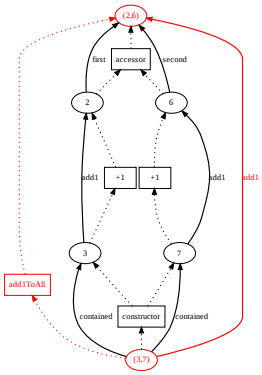


Figure 5: Overlapping and Hierarchical Accounts in a Provenance Graph

the process to take place. On the other hand, since the edges "was generated by" from artifact (3,7) are colored differently, they indicate that alternate explanations exist for the process that led to such artifact being as it is.

It is possible to use refinements repeatedly to create a hierarchy of accounts, as illustrated in Figure 6. We see that a third account (blue) is introduced, to explain how one of the +1 processes was performed.

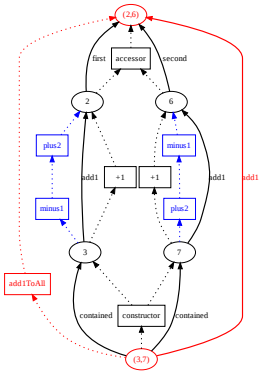


Figure 6: Hierarchy of Accounts in a Provenance Graph

By combining several accounts, we can obtain cycles, as illustrated by Figure 7 (left). Here, in the first account (darker, black), a description of two processes p1a and p1b is presented, and their dependencies on artifacts a0, a1, a2 and a3. In the second account (lighter, red), it is stated that the two processes p1a and p1b constitute a single process operating on inputs a0 and a2, and producing a1 and a3. If we combine the two views, a cycle of "used" and "was generated by" edges has been created: a2 → p2 → a1 → p1 → a2. In the right-hand side of Figure 7, we make data derivations explicit: in this example, we observe that no cycle of "was derived from" is created, since the two accounts are compatible (since one provides

more details than the other). In the most general case, where accounts may be conflicting, we can anticipate cycle of "was derived from" edges to be resulting from the union of several accounts.

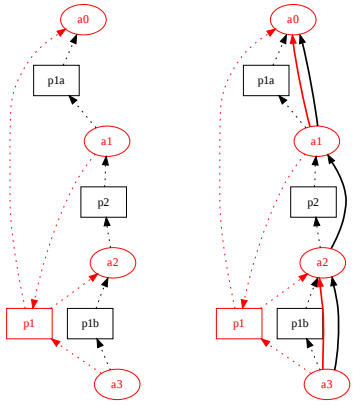


Figure 7: Multiple Accounts Creating Cycle: without (left) and with (right) "was derived from" edges

While overlapping accounts are intended to allow various descriptions of a same execution, it is recognized that these accounts may differ in their description's semantics. In general, such semantic differences may not be expressed by structural properties we can set constraints on in the model (beyond the constraints identified in this document).

5. Temporal Constraints and Observation Time

The Open Provenance Model allows for causality graphs to be decorated with time information. In this model, time is *not* intended to be used for deriving causality: if causal dependencies exist, they need to be made explicit with the appropriate edges. OPM is compatible with causality in distributed systems [16]: when an effect event is caused by a cause event, then the cause "happened before" the effect (where "happened before" is Lamport's partial ordering [16]). Furthermore, given that time may have been observed by an observer, we would expect such time information to be compatible with causal dependencies. If a same clock is used to measure time for both the effect and cause, then the time of an effect should be greater than the time of its cause. Hence, time is useful in validating causality claims.

In the Open Provenance Model, time may be associated to *instantaneous occurrences* in a process. We currently recognize four instantaneous occurrences, which have a reasonable shared understanding in real life and computer systems. Two of them pertain to artifacts, whereas the other two relate to processes. For artifacts, we consider the occurrences of *creation* and *use*, whereas for processes, we consider their *starting* and *ending*.

The rationale for choosing instantaneous time for the OPM model is the same as for adopting artifacts as immutable pieces of state. At a specific time, an object we consider was in a specific state, which we refer to as artifact, and for which we can express the causality path that led to the object being in such a state.

In some scenarios, occurrences of use or creation of objects and occurrences of starting or ending of processes may not be instantaneous. To capture such scenarios, detailed processes and artifacts, and their respective causal dependencies, need to be made explicit, in order to be expressible in the OPM model. For instance, the starting of a nuclear power plant or of a job scheduling activity is not usefully modelled as an instantaneous occurrence, when one tries to understand failures that occurred during this activity; hence, this whole starting occurrence must be modelled by one process (or possibly several), which in turn have instantaneous beginnings and endings.

In the Open Provenance Model, time information is expected to be acquired by an observer's *observing* a clock[2] when an occurrence occurs. Given that time is observed, time accuracy is limited by the granularity of the clock and the granularity of the observer's activities. Hence, while the notion of time we consider is instantaneous, the model allows for an interval of accuracy to support granularity of clocks and observers. In the OPM model, an instantaneous occurrence happening at time t is specified in term of two observation times t^m, t^M, such that the occurrence is known to have occurred *no later* than t^M and no earlier than t^m. Hence, $t \in [t^m, t^M]$.

Concretely, for an artifact, we will be able to state that it was used (or generated by) no earlier than time t_1 and/or no later than time t_2. For a process, we will be able to state that it was started (or terminated), no earlier than time t_1 and/or no later than time t_2.

In Figure 8, we revisit OPM entities indicating how time information may be expressed in the model. We note again that time information is optional in OPM and is expressed as an observation time interval.

Edges "used" and "was generated by" can be extended with an *optional* timestamp, indicating that the associated artifact was known to be generated or used, at a given time.

For a "was controlled by" edge, we allow two *optional* timestamps marking when the process was known to be started or terminated, respectively. In a given account, for a process that is not source of a "was controlled by" edge, we allow the process to be decorated by two timestamps directly.

For a "was derived from" edge, one *optional* timestamp is permitted, which indicates when the artifact was used. Likewise, for "was triggered by" edge, we also allow one *optional* timestamp that marks the time when the communicated artifact was used by the edge source.

[2] OPM assumes that all clocks are properly synchronized.

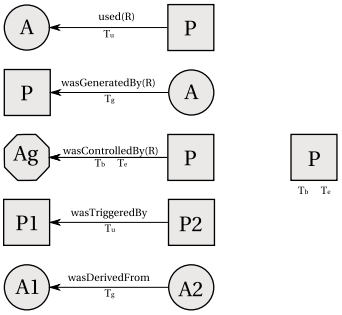
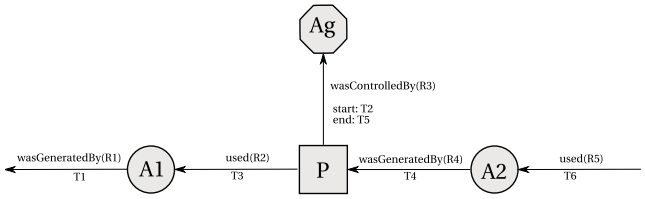


Figure 8: Time in the Provenance Model

The model of causality in OPM is essentially timeless since time precedence does not imply causality: if a process P_1 "happened before" a process P_2, in general, we cannot infer that P_1 caused P_2 to happen. However, the converse implication holds; furthermore, assuming time is measured according to a single clock (or synchronized clocks), time observations will be comparable.



$T1 \leq T3$ (artifact A1 must exist before being used)
$T2 \leq T3$ (process P must have started before using artifact A1)
$T3 \leq T5$ (process P uses artifact A1 before it ends)
$T2 \leq T4$ (process P must have started before generating artifact A2)
$T4 \leq T5$ (process P generates artifact A2 before it ends)
$T4 \leq T6$ (artifact A2 must exist before being used)
$T2 \leq T5$ (process P must have started before ending)
no constraint between T3 and T4

Figure 9: Relation "Happened Before" in the Open Provenance Model

Figure 9 displays the various "happened before" relationship that must be satisfied in OPM. We write $T_1 \leq T_3$ to express that the event observed at time T_1 happened before the event observed at time T_3. When the two time observations are made with the same clock (or synchronized clocks), then time observations can be compared. According to Figure 9, an artifact must exist before it is being used ($T_1 \leq T_3$ and $T_4 \leq T_6$). If an artifact is used by a process, it will actually be used after the start of the process ($T_2 \leq T_3$) and before the end of the process ($T_3 \leq T_5$). A process generates artifacts before its end ($T_4 \leq T_5$), and a process starts precedes its generation of artifacts ($T_2 \leq T_4$) and its end ($T_2 \leq T_5$).

6. Completion and Inferences

The Open Provenance Model has defined the notion of *OPM graph* based on a set of syntactic rules and topological constraints. Provenance graphs are aimed at representing causality graphs explaining how processes and artifacts came out to be. It is expected that a variety of reasoning algorithms will exploit this data model, in order to provide novel and powerful functionality to users. It is beyond the scope of this document to include an extensive coverage of relevant reasoning algorithms. However, provenance graphs, by means of edges, capture causal dependencies, which can be summarized by means of transitive closure that we describe in this section. First, we introduce completion rules, and then define multi-step inferences.

6.1. Completion Rules

In Section 3, we have introduced the two causal dependencies "was triggered by" and "was derived from" as summary edges for a process view (where an intermediary artifact was unknown) and a data view (where an intermediary process was unknown), respectively. Figures 10 and 11 describe *completion rules*, i.e. one-step transforms that can be performed in the Open Provenance Model. A rule explains how a subgraph can be converted into another subgraph.

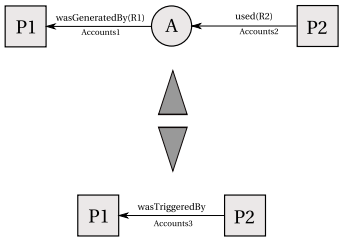


Figure 10: Completion: Artifact Introduction and Elimination

Figure 10 displays a bidirectional transformation. According to the forward transformation (referred to as *artifact elimination*), a "was triggered by" edge can be obtained from the existence of "used" and "was generated by" edges. We note that the novel "was triggered by" edge belongs to the set of accounts given by the intersection[3] of accounts of the "used" and "was generated by" edges.

Figure 10 shows a bidirectional completion rule: *artifact introduction* allows us to establish that the "was triggered by" edge is hiding the existence of some artifact used by P_2 and generated by P_1. The novel edges

[3] Taking the intersection of accounts ensures that the edges derived by artifact elimination are meaningful in the account they are declared to be member of. Furthemore, this ensures that completion rules preserve the effective account membership of all nodes in the graph.

"used" and "was generated by" are asserted in the same account context as the original "was triggered by" edge. The completion rule allows us to establish the existence of some artifact but it does not tell us what their id is. This is the consequence of using "was triggered by", which is a lossy summary of the composition of "used" and "was generated by".

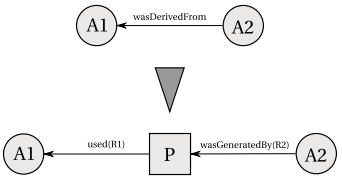


Figure 11: Completion: Process Introduction

In Figure 11, there is only one completion rule, referred to as *process introduction*: a "was derived from" edge hides the presence of an intermediary process. Novel edges are asserted with the same accounts as the original edge. The converse rule does not hold however, since, without any internal knowledge of P, it is impossible[4] to ascertain there is an *actual* dependency between A_1 and A_2.

6.2. Multi-Step Inferences

When users want to find out the causes of an artifact or a process, they may not just be interested in direct causes, but in indirect causes, as well, involving multiple transitions. Hence, for the purpose of expressing queries or expressing inferences about provenance graphs, we introduce *four new relationships*, which are multi-step versions of existing relationships. We first introduce the multi-step "was derived from" relation, from which other versions are obtained.

Definition 11 (Multi-Step WasDerivedFrom). *An artifact a_1 was derived from a_2 (possibly using multiple steps), written as $a_1 \rightarrow^* a_2$, if a_1 "was derived from" an artifact that was a_2 or that was itself derived from a_2 (possibly using multiple steps). In other words, it is the transitive closure of the edge "was derived from". It expresses that artifact a_2 had an influence on artifact a_1.*

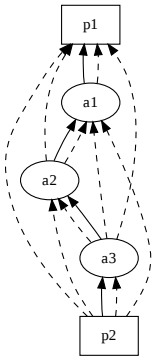
From Definition 11, we formulate convenience multi-step relations as follows.

Definition 12 (Secondary Multi-Step Edges).

- *Process p used artifact a (possibly using multiple steps), written $p \rightarrow^* a$, if p used an artifact that was a or was derived from a (possibly using multiple steps).*
- *Artifact a was generated by process p (possibly using multiple steps), written $a \rightarrow^* p$, if a was an artifact or was derived from an artifact (possibly using multiple steps) that was generated by p.*
- *Process p_1 was triggered by process p_2 (possibly using multiple steps), written $p_1 \rightarrow^* p_2$, if p_1 used an artifact that was generated or was derived from an artifact (possibly using multiple steps) that was itself generated by p_2.*

Intuitively, multi-step edges can be inferred from single-step edges, by "eliminating" artifacts that occur in chains of dependencies (Note that inferences do not allow process elimination.)

The four relationships, and associated inferences, are illustrated in Figure 12. In this figure, plain edges represent single-step dependencies, whereas dashed edges represent multi-step dependencies. For instance, from $p2 \rightarrow a3 \rightarrow a2$ we can infer $p2 \rightarrow^* a3 \rightarrow^* a2$ and $p2 \rightarrow^* a2$, by "eliminating" $a3$.



Single-Step Edges (plain)

$GeneratedBy$: $a1 \rightarrow p1$
$Used$: $p2 \rightarrow a3$
$DerivedFrom$: $a3 \rightarrow a2,\ a2 \rightarrow a1$

Inferrable Multi-Step Edges (dashed)

$GeneratedBy^*$: $a1 \rightarrow^* p1, a2 \rightarrow^* p1, a3 \rightarrow^* p1$
$Used^*$: $p2 \rightarrow^* a3, p2 \rightarrow^* a2, p2 \rightarrow^* a1$
$DerivedFrom^*$: $a3 \rightarrow^* a2, a3 \rightarrow^* a1, a2 \rightarrow^* a1$
$TriggeredBy^*$: $p2 \rightarrow^* p1$

Figure 12: Inference: Multi-Step Edges

7. Provenance Graph Definition

We assume the existence of a few primitive sets: identifiers for processes, artifacts and agents, roles, and accounts. These sets of identifiers provide identities to the corresponding entities within the scope of a given provenance graph. A given serialization will standardize on these sets, and provide concrete representations for them.

It is important to stress that the purpose of identifiers is to define the structure of graphs: they are not meant to

[4] It is suggested that a profile could offer an annotation indicating that all outputs of a process are dependent on all its inputs. For processes annotated in this way, the converse inference, i.e. process elimination, would hold.

define identities that are persistent and reliably resolvable over time.

The open provenance model is defined according to the following rules.

1. An OPM *entity* can be a node, an edge, a role, an account, or a graph.
2. *Accounts* are identified by unique identifiers. An account represents a description at some level of detail as provided by one or more observers. Two accounts are equal if and only if they have the same identifier.
3. *Artifacts* are identified by unique identifiers. Artifacts are entities that represent an application instantaneous piece of state. Two artifacts are equal if and only if they have the same identifier (irrespective of the state they represent[5]). Artifacts can optionally belong to accounts: account membership is declared by listing the accounts an artifact belongs to.
4. *Processes* are identified by unique identifiers. Processes represent applications activities. Two processes are equal if and only if they have the same identifier. Processes can optionally belong to accounts: account membership is declared by listing the accounts a process belongs to.
5. *Agents* are identified by unique identifiers. Agents represent contextual entities controlling processes. Two agents are equal if and only if they have the same identifier. Agents can optionally belong to accounts: account membership is declared by listing the accounts an agent belongs to.
6. *Edges* are identified by their source, destination, and role (for those that include a role). Edges represent causal dependencies between their source (the effect) and their destination (the cause). The source and destination consist of identifiers for artifacts, processes, or agents, according to Figure 1. Edges can also optionally belong to accounts: account membership is defined by listing the accounts an edge belongs to. Structural equality applies to edges: two edges of type "used" (resp. "was generated by", or "was controlled by") are equal if they have the same source, the same destination, the same role, and the same accounts; two edges of type "was derived from" (resp. "was triggered by") are equal if they have the same source, the same destination, and the same accounts. The meaning of roles is not defined by OPM but by application domains; OPM only uses roles syntactically (as "tags") to distinguish the involvement of artifacts and agents in processes.
7. Roles are mandatory in edges "used", "was generated by", and "was controlled by". The meaning of a role is defined by the semantics of the process they relate to. Role semantics is beyond the scope of OPM.
8. To ensure that edges establish a causal connection between actual causes and effects, the model assumes that if an edge belongs to an account, then its source and destination also belong to this account. In other words, the *effective account membership* of an artifact/process/agent is its declared account membership and the account membership of the edges it is adjacent to (i.e., it is source and destination of).
9. An *OPM graph* consists of artifacts, processes, agents, edges, and accounts, as specified above. OPM graphs may be disconnected. OPM graphs can be compared by using structural equality. The empty set is an OPM graph. A singleton containing an artifact, a process or an agent is an OPM graph. The set of OPM graphs is closed under the intersection and union operations[6], i.e. the intersection of two OPM graphs is an OPM graph (and likewise for union). We note at this stage that syntactically valid OPM graphs may not necessarily make sense from a provenance viewpoint.
10. A view of an OPM graph according to *one* account, referred to as *account view*, consists of elements whose effective account membership for artifacts, processes, and agents, and account membership for edges contain the account.
11. While cycles can be expressed in the syntax of OPM, an account view is *legal* if it is free of cycle of "was derived from" edges and if it contains at most one "was generated by" edge per artifact. This ensures that within one account, an OPM graph captures proper causal dependencies, and that a single explanation of the origin of an artifact is given.
12. Hence, a *legal OPM graph* is one for which all account views are legal.
13. Legal account views are OPM graphs. The union of two legal account views is an OPM graph (it is not necessarily a legal view since it may contain cycles). The intersection of two legal account views is a legal account view.
14. A provenance graph is not required to contain time information.
15. *Edges* can optionally be decorated with time information (as per Figure 9). In a given account, a *Process* without "was controlled by" edge can also optionally be decorated with time information.
16. Within an account, time information must be consistent with causality. To this end, the definition of

[5] In the Open Provenance Model, artifact identifiers are the only way to distinguish artifacts in the graph structure. Two artifacts differ if they have different ids, even though they may refer to a same application data product. Two different artifacts are therefore separate nodes in a provenance graph: they have two different computational histories.

[6] Equality, union and intersection of OPM graphs require a predicate to be provided allowing nodes and edges to be compared *across* graphs. For instance, such a predicate can make use of the global, persistent name (`pname`) introduced in Section 8.2.

legality of an account view is extended with an extra condition requiring that *causation is time-monotonic*, as displayed in Figure 9 (for identical or synchronized clocks) .
All observed times are pairs of instantaneous time values. For $T_1 = (t_1^m, t_1^M)$, with $t_1^m \leq t_1^M$, and $T_2 = (t_2^m, t_2^M)$, with $t_2^m \leq t_2^M$ inequality is defined as follows: $T_1 \leq T_2$ if $t_1^m \leq t_1^M \leq t_2^m \leq t_2^M$.

17. Two account views are said to be *overlapping* if the views have some artifact, process or agent in common[7].
18. An account view v_1 is a *refinement* of another account view v_2 if the set of multi-step dependencies that can inferred in v_1 after application of completion rules is a superset of multi-steps dependencies that can be inferred in v_2 after application of completion rules.
19. In an OPM graph, relations between accounts (overlap, refinement, and any other) may be asserted. Account relation assertions are legal if two account asserted to be in relationship satisfy this relationship's definition.

8. Annotations

Practical experience with the third Provenance Challenge has shown the need for "extra information" to be added to OPM entities. Such extra information is typically required for inter-operability purpose, to allow meaningful exchange of provenance information. Examples include subtyping of edges, descriptions of processes, and reference to values of artifacts. To accommodate "extra information" in an extensible manner, the Open Provenance Model allows for all its entities to be annotated, by means of the OPM annotation framework, which we describe below.

8.1. The OPM Annotation Framework

The OPM annotation framework is defined according to the following rules.

1. *OPM annotations* are also OPM entities, forming a class of objects distinct from the other OPM entities.
2. An *annotable entity* can be an OPM graph[8], an OPM node, an OPM edge, an OPM account, an OPM role, or an OPM annotation.
3. An *annotated entity* is an annotable entity associated with one or more instances of annotations.
4. Every annotated entity must be uniquely identifiable in the context of an OPM graph by means of an identifier.
5. An annotation instance is an object of the class OPM Annotation and consists of the following:
   - a subject: an annotable entity (identified by its identifier) to which the annotation is attached;
   - a non-empty set of property-value pairs:
     - the property includes a namespace to represent its scope,
     - the value must be typed;
   - a list of accounts, which must be a subset of the effective accounts of the annotated entity.

   The intended *meaning* of a property-value pair is that the annotated entity (i.e. the subject) is provided with additional descriptions, each consisting of a property of the subject and the value of this property for the subject, in the context of some accounts. Multiple property-value pairs are allowed within an annotation instance. It is legal for a same property to occur multiple times with different values.
6. Annotations can themselves be annotated and subtyped.

Figure 13 illustrates how annotations have been added to Figure 3. We have two[9] annotations represented as a "post-it", with property "quality" and value "yummy" for the cake, and property "type" and value "raising" from flour. Also the edges "was derived from" were subtyped, and their type added as a label.

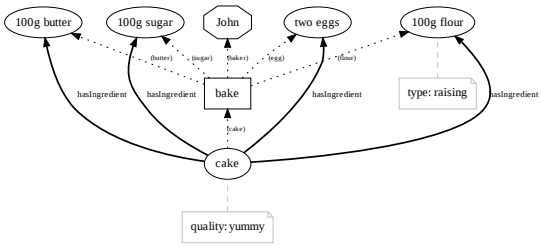


Figure 13: Annotation in the Victoria Sponge Cake Provenance

8.2. Common OPM Properties

For inter-operability purpose, OPM defines a set of common properties. We identify each property by a unique URI; we define the expected type of subjects and values associated with such property. Finally, we state the intended meaning of the property.

[7] Whilst one could infer whether two graphs actually overlap, this would typically require the graphs to be parsed fully in order to make such an inference; instead, explicit declarations of such overlapping properties can be considered to facilitate the processing and traversal of graphs.

[8] OPM is intended to be technology agnostic. However, there is an acknowledgement that annotating a graph may present challenges with some technologies such as RDF. The implications of such capability are currently under investigation.

[9] In fact, all nodes and edges are annotated because they all have a label. We did not make the "label" annotation explicit in the graphical representation of annotations since the label is already displayed in nodes and along side edges.

type	subject:	an annotable entity
	property:	`http://openprovenance.org/property#type`
	value:	*a URI*
	meaning:	*Denotes the subtype of an OPM entity. Such subtypes are represented by a URI.*
pname	subject:	an annotable entity
	property:	`http://openprovenance.org/property#pname`
	value:	*a URI*
	meaning:	*Denotes a persistent name that can be used by OPM graph queriers to compare OPM entities across graphs. The scope of this name is intended to be global.*
label	subject:	an annotable entity
	property:	`http://openprovenance.org/property#label`
	value:	*a String*
	meaning:	*This property provides a human-readable version of an OPM entity.*
value	subject	an artifact
	property:	`http://openprovenance.org/property#value`
	value:	*a typed value*
	meaning:	*Denotes a serialization of an application value associated with an OPM entity. Such serialization should have a type (expressed in a type system suitable for the serialization). Serialization technologies include XML, JSON, and ntriples.*
encoding	subject:	an artifact or an OPM graph
	property:	`http://openprovenance.org/property#encoding`
	value:	*a URI*
	meaning:	*Denotes how a serialization was constructed. For instance, using the Java bean serializer to create an XML document, by applying a specified transformation to the application data, e.g. anonymisation, by passing a reference to the actual value, or by creating a set of RDF triples.*
profile	subject:	an OPM graph
	property:	`http://openprovenance.org/property#profile`
	value:	*a URI*
	meaning:	*This property applies to an OPM graph and denotes a profile that is supported by that graph*

9. OPM Profiles

OPM is a top-level representation framework for provenance, and we recognize that some communities will develop their own best practice and usage guidelines. To encourage such a notion of best practice or usage guideline, we formalize it by means of the concept of an *OPM profile*. For instance, a set of conventions is currently emerging to represent "collections" in OPM; it is suggested that all these conventions can be expressed in a "collection profile" [17]. Whenever an OPM graph adopts these conventions, it can be annotated with this profile so that queriers may exploit this declaration in order to process the graph.

An OPM profile is intended to define a specialisation of OPM, and therefore must remain compatible with the semantics of OPM described in this document. Concretely, this means that a profile-compliant OPM graph *is* an OPM graph, whose semantics is described in this document. This implies that all inferences specified by this document remain valid in a profile-compliant OPM graph. For the avoidance of doubt, any extension of OPM that does not preserve the OPM semantics must not be defined as a profile, and must not be referred to as OPM. Profiles are specified in separate documents that are independent of this core specification.

An OPM profile consists of the following elements:

1. A mandatory unique global identifier for the profile. Such a profile identifier must be used as the value of the profile property in an annotation to the OPM graph that supports such a profile.
2. An optional controlled vocabulary for annotations. In this context, a controlled vocabulary for annotations is a specification of the properties, its permitted subjects, and its permitted values (such as types or enumerated values). Such a controlled vocabulary may be used for some of the following:
   (a) Subtyping edges and nodes in OPM graphs by means of the type property;
   (b) Defining application-specific properties: for instance, a position property attached to nodes can be exploited by a visualization tool to render OPM graphs.
3. Optional general guidance to express OPM graphs. There are typically many different ways in which OPM can be used to describe an execution. For inter-operability purpose, it is therefore good to provide some guidance on how to structure OPM graphs. For instance, it may be useful to identify several types of accounts (e.g., for high-level and low-level descriptions) and to mandate that each account contains edges of specific subtypes.
   Likewise, common software engineering patterns involved in the design and implementation of an application may also be reflected in OPM graphs; for instance, the publish/subscribe pattern of an application can result in a set of OPM conventions to express publisher and consumer processes and the flow of information between them.
4. Optional profile expansion rules.
   In some specific circumstances, it may not be necessary to express all edges or nodes related to an execution because they can be derived. Hence, profiles may contain rules, referred to as *expansion rules* to convert a profile-compliant OPM graph into another OPM graph. The process of applying profile expansion rules to generate an OPM graph is called *profile expansion*, and the resulting graph is said to be *profile-expanded*. We draw the reader's attention to the terminology adopted here. Profile expansion should be distinguished from the completion rules and multi-step inferences defined in Section 6.
   Profile expansion constructs a profile-expanded OPM graph by adding new elements (and possibly removing some), satisfying the following constraints:
   (a) A profile-compliant graph is an OPM graph;
   (b) A profile-expanded graph is an OPM graph,
   (c) The semantics of the profile-compliant graph and of the profile-expanded graph are solely defined by this document;
   (d) Any multi-step edge that can be inferred between two nodes in a profile-compliant graph must also be inferable in the profile-expanded

graph (but not vice-versa)[10]

(e) Provided that condition (4d) holds, the profile expansion process is:
- node preserving: any node in the profile-compliant graph also belongs to the profile-expanded graph;
- single-step edge lossy: single-step edges in the profile-compliant graph may not necessarily belong to the profile-expanded graph;
- multip-step edge preserving: multi-step edges that can be inferred in the profile-compliant graph must also be inferrable in the profile-expanded graph;
- annotation lossy: profile-specific annotations in the profile-compliant graph may not necessarily belong to the profile-expanded graph.

As a result, there is not need of knowing about a profile to be able to analyse a profile-expanded graph. From a reasoning perspective, an OPM reasoning engine is only required to implement the inference rules described in this document. Profile-compliant OPM graphs can be translated into OPM graphs by the profile expansion process. Alternatively, a reasoning engine may be profile aware, and may be able to reason on profile-compliant OPM graphs without requiring profile expansion to take place.

5. Optional serialization specific syntax.
A profile may introduce syntactic short-cuts for specific serializations. The serialization needs to explain how such short-cuts can be translated into core OPM, and vice-versa.

We can envisage that controlled vocabularies, patterns and inference rules may all be expressed in some declarative language, which could be used to automatically check whether an OPM graph is compliant with a profile, and to perform profile expansion automatically. There is however no off-the-shelf solution that we can reuse for this purpose. Hence, our assumption is that profiles will be mostly specified in natural language, and that profile compliance and profile expansion routines will have to be implemented by hand. We welcome solutions to make these steps as automatic as possible.

10. Discussion, Related and Future Work

OPM addresses the requirements identified in Section 2. Fourteen teams participating in the Third Provenance Challenge have demonstrated that OPM can be used to exchange provenance information. Common tools are emerging (see openprovenance.org), such as visualization and conversion, some of which were demonstrated in the Third Provenance Challenge (see papers in this special issue).

This specification defines the Open Provenance Model in a technology-agnostic manner, and is used to generate the provenance of data products produced using multiple technologies (e.g., C#, Java, Kepler, Taverna, PASS, VisTrails). The specification also defines the kind of inferences that are permitted; they can be classified in three categories: completion (Section 6.1), multi-step inference (Section 6.2) and profile expansion (Section 9). The concept of account allows multiple descriptions to coexist. Finally, the cake example, though contrived, illustrates that OPM can be applied to physical artifacts. OPM is described as an abstract model, but serializations to XML and RDF (and associated XML Schema and OWL ontology) are being proposed (openprovenance.org) and have actively been used in the Third Provenance Challenge.

Prior to the first OPM specification, multiple provenance technologies had been developed, but none aimed at defining a technology-agnostic provenance data model for inter-operability purpose. For instance, PASOA [18] offers a model that aimed at inter-operability between execution technologies: it focuses on distribution (message-passing systems) and its definition is bound to XML. So, OPM is the first model to be purely technology agnostic. A companion paper [19] defines its formal semantics. Since the conception of OPM, other models have emerged. Hartig [20] proposes the *provenance vocabulary*, which we conjecture can be defined as a profile of OPM, to describe the provenance of Linked Data over the Web. His model accounts for the creation and access of RDF data, and is strongly bound to RDF technology. Sahoo *et al.* [21] define a provenance ontology based on three entities similarly to OPM, but their design is influenced by scientific experiments; their analogous of artifact denotes potentially stateful electronic data (including collections which OPM defines in a separate profile). In addition, relationships between entities are not all causal.

The Proof Markup Language (PML) [22], conceived independently in the context of the Semantic Web, includes metadata such as authorship and authoritativeness of a source, and a detailed trace of inference rules applied. Relationships, which capture notions of Consequent and Antecedents to a proof step, the succession of which consists of a proof, bear some strong similarity with OPM concepts.

The W3C Incubator on Provenance [24] has identified use cases and requirements for provenance on the Web, and is proposing a mapping of the above models of provenance to OPM.

OPM is a language to describe dependencies between artifacts, processes, and agents. Since the Third Provenance Challenge did not test agents much, further guidance is needed on how best to describe systems in the presence of agents. For instance, in the OPM 1.01 specification [12], we identified alternate patterns by which agents controlled processes, according to different accounts. Further work is required to develop profiles, based on commu-

[10] In fact, the profile expansion rules generate an OPM graph that is a refinement of the original graph. Any node of the profile-compliant graph is also a node of the profile-expanded graph (but the latter may contain extra nodes). Any multi-step edge that can be inferred in the profile-compliant graph can also be inferred in the profile-expanded graph.

nity experience with these OPM constructs.

Scientists regularly manipulate sets of data as first-class entities. While such sets, referred to as collections, can be represented in OPM as artifacts, their provenance is typically tightly linked to the provenance of their constituents. However, no guidance is provided by OPM to express such collections and their relation to their constituents. To acknowledge the importance of collections, a whole section on collections was introduced in OPM 1.01. Since then, the concept of profile has been formulated, and a collection profile has been drafted [17]. During the design phase of OPM v1.1, a vote unanimously opted to keep the collection profile separate from OPM core. Furthermore, OPM considers artifacts as immutable pieces of state; guidance is required to represent stateful objects in OPM.

OPM does not provide any specific mechanism to assert attribution of a provenance graph or portion thereof. It is generally recognized that annotations are the mechanism to do so; attribution could be attached, as an annotation, to accounts or to the graph itself, for example. Work is underway to define a Dublin Core profile for OPM [23], which deals with some of these concepts.

11. Conclusion

The document has introduced the Open Provenance Model, consisting of a technology-independent specification and a graphical notation, to express causality graphs representing *past* executions. Work is in progress to define several useful profiles, such as the Dublin Core and the Collections profiles, specify serialization formats to XML and RDF, and formalize the OPM semantics. We will also specify protocols by which provenance of entities can be queried, and protocols for applications to record descriptions of their execution.

12. Acknolwedgement

The authors of this document gratefully acknoledge the contributions made by authors of previous versions of the specification Roger Barga, Shawn Bowers, Tommy Ellkvist, Carole Goble, Bertram Ludaescher, Robert E. McGrath, and Patrick Paulson.

[1] P. W. Group, Data Dictionary for Preservation Metadata — Final Report of the PREMIS Working Group, Tech. Rep., Preservation Metadata: Implementation Strategies (PREMIS), URL http://www.oclc.org/research/projects/pmwg/premis-final.pdf, 2005.

[2] Y. Simmhan, B. Plale, D. Gannon, A survey of data provenance in e-science, SIGMOD Record 34 (3) (2005) 31–36, URL http://www.sigmod.org/sigmod/record/issues/0509/p31-special-sw-section-5.pdf.

[3] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, J. Myers, Examining the Challenges of Scientific Workflows, IEEE Computer 40 (12) (2007) 26–34, doi:http://doi.ieeecomputersociety.org/10.1109/MC.2007.421, URL http://www.ecs.soton.ac.uk/~lavm/papers/computer07.pdf.

[4] L. Moreau, I. Foster (Eds.), Provenance and Annotation of Data — International Provenance and Annotation Workshop, IPAW 2006, vol. 4145 of *Lecture Notes in Computer Science*, Springer-Verlag, ISBN 3-540-46302-X, URL http://www.springer.com/uk/home/generic/search/results?SGWID=3-40109-22-173681711-0, 2006.

[5] R. Bose, I. Foster, L. Moreau, Report on the International Provenance and Annotation Workshop (IPAW06), Sigmod Records 35 (3) (2006) 51–53, ISSN 0163-5808, doi:http://doi.acm.org/10.1145/1168092.1168102, URL http://www.sigmod.org/sigmod/record/issues/0609/sigmod-record.september2006.pdf.

[6] L. Moreau, B. Ludäscher, I. Altintas, R. S. Barga, S. Bowers, S. Callahan, G. Chin Jr., B. Clifford, S. Cohen, S. Cohen-Boulakia, S. Davidson, E. Deelman, L. Digiampietri, I. Foster, J. Freire, J. Frew, J. Futrelle, T. Gibson, Y. Gil, C. Goble, J. Golbeck, P. Groth, D. A. Holland, S. Jiang, J. Kim, D. Koop, A. Krenek, T. McPhillips, G. Mehta, S. Miles, D. Metzger, S. Munroe, J. Myers, B. Plale, N. Podhorszki, V. Ratnakar, E. Santos, C. Scheidegger, K. Schuchardt, M. Seltzer, Y. L. Simmhan, C. Silva, P. Slaughter, E. Stephan, R. Stevens, D. Turi, H. Vo, M. Wilde, J. Zhao, Y. Zhao, The First Provenance Challenge, Concurrency and Computation: Practice and Experience 20 (5) (2008) 409–418, ISSN 1532-0626, doi:DOI:10.1002/cpe.1233, URL http://www.ecs.soton.ac.uk/~lavm/papers/challenge-editorial.pdf.

[7] Second:Challenge, Second Challenge Team Contributions, URL http://twiki.ipaw.info/bin/view/Challenge/ParticipatingTeams, 2007.

[8] L. Moreau, J. Freire, J. Futrelle, R. E. McGrath, J. Myers, P. Paulson, The Open Provenance Model (v1.00), Tech. Rep., University of Southampton, URL http://eprints.ecs.soton.ac.uk/14979/1/opm.pdf, 2007.

[9] S. Miles, Technical Summary of the Second Provenance Challenge Workshop, Tech. Rep., King's College, URL http://twiki.ipaw.info/bin/view/Challenge/SecondWorkshopMinutes, 2007.

[10] First:OPM:Workshop, Open Provenance Model Workshop: Towards Provenance Challenge 3, URL http://twiki.ipaw.info/bin/view/Challenge/OpenProvenanceModelWorkshop, 2008.

[11] P. Groth, First OPM Workshop Minutes, Tech. Rep., Information Science Institute, USC, URL http://twiki.ipaw.info/bin/view/Challenge/FirstOPMWorkshopMinutes, 2008.

[12] L. Moreau (Editor), B. Plale, S. Miles, C. Goble, P. Missier, R. Barga, Y. Simmhan, J. Futrelle, R. McGrath, J. Myers, P. Paulson, S. Bowers, B. Ludaescher, N. Kwasnikowska, J. Van den Bussche, T. Ellkvist, J. Freire, P. Groth, The Open Provenance Model (v1.01), Tech. Rep., University of Southampton, URL http://eprints.ecs.soton.ac.uk/16148/1/opm-v1.01.pdf, 2008.

[13] L. Moreau, J. Freire, J. Futrelle, J. Myers, P. Paulson, Governance of the Open Provenance Model, URL http://twiki.ipaw.info/pub/OPM/WebHome/governance.pdf, 2009.

[14] OPM:twiki, Open Provenance Model Wiki, URL http://twiki.ipaw.info/bin/view/OPM/, 2009.

[15] J. Woodcock, J. Davies, Using Z. Specification, Refinement, and Proof, Prentice Hall, ISBN 0139484728, 1996.

[16] L. Lamport, Time, Clocks, and the Ordering of Events in a Distributed System, Communications of the ACM 21 (7) (1978) 558–565, doi:http://doi.acm.org/10.1145/359545.359563, URL http://research.microsoft.com/en-us/um/people/lamport/pubs/time-clocks.pdf.

[17] P. Groth, S. Miles, P. Missier, L. Moreau, A Proposal for Handling Collections in the Open Provenance Model, URL http://mailman.ecs.soton.ac.uk/pipermail/provenance-challenge-ipaw-info/2009-June/000120.html, 2009.

[18] P. Groth, S. Miles, L. Moreau, A Model of Process Documentation to Determine Provenance in Mash-ups, Transactions on Internet Technology (TOIT) 9 (1) (2009) 1–31, ISSN 1533-5399, doi:http://doi.acm.org/10.1145/1462159.1462162, URL http:

//www.ecs.soton.ac.uk/~lavm/papers/toit09.pdf.

[19] L. Moreau, N. Kwasnikowska, J. Van den Bussche, The Foundations of the Open Provenance Model, Tech. Rep., University of Southampton, URL http://eprints.ecs.soton.ac.uk/17282/, 2009.

[20] O. Hartig, Provenance Information in the Web of Data, in: Proceedings of the Linked Data on the Web Workshop (LDOW'09), Madrid, Spain, URL http://events.linkeddata.org/ldow2009/papers/ldow2009_paper18.pdf, 2009.

[21] S. S. Sahoo, A. Sheth, C. Henson, Semantic Provenance for eScience: Managing the Deluge of Scientific Data, Internet Computing, IEEE 12 (4) (2008) 46–54, ISSN 1089-7801, doi: http://dx.doi.org/10.1109/MIC.2008.86.

[22] D. L. McGuinness, P. Pinheiro da Silva, Explaining answers from the Semantic Web: the Inference Web approach, J. Web Sem. 1 (4) (2004) 397–413, doi:http://dx.doi.org/10.1016/j.websem.2004.06.002, URL http://ksl.stanford.edu/KSL_Abstracts/KSL-04-03.html.

[23] S. Miles, L. Moreau, J. Futrelle, OPM Profile for Dublin Core Terms (Draft), URL http://mailman.ecs.soton.ac.uk/pipermail/provenance-challenge-ipaw-info/2009-June/000124.html, 2009.

[24] xg-prov, W3C Provenance Incubator Group Wiki, http://www.w3.org/2005/Incubator/prov/, 2010.