The Open Provenance Model

Luc Moreau, University of Southampton,
Juliana Freire, University of Utah,
Joe Futrelle, NCSA,
Robert E. McGrath, NCSA
Jim Myers, NCSA,
Patrick Paulson, PNNL

December 18, 2007

1 Introduction

Provenance is well understood in the context of art or digital libaries, where it respectively refers to the documented history of an art object, or the documentation of processes in a digital object's life cycle [3]. Interest for provenance in the "e-science community" [9] is also growing, since provenance is perceived as a crucial component of workflow systems [2] that can help scientists ensure reproducibility of their scientific analyses and processes.

Against this background, the *International Provenance and Annotation Workshop* (IPAW'06), held on May 3-5, 2006 in Chicago, involved some 50 participants interested in the issues of data provenance, process documentation, data derivation, and data annotation [5, 1]. 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 was set up in order 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 inter-operability of systems, by exchanging provenance information. Thirteen teams [8] responded to this second challenge. Discussions indicated that there was substantial agreement on a core representation of provenance. As a result, in a workshop on August 7-8 in Salt Lake City, the authors met, and crafted and iterated a data model, which is presented in this paper.

The starting point of this work is the community agreement summarized by Miles [4]. 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*, and not a description of something that could happen in the future.

In this paper, we introduce the *Open Provenance Model*, a model for provenance which meets 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 the model in a precise, technology-agnostic manner.
- To support a digital representation of provenance for any "thing", whether produced by computer systems or not.
- To define a core set of rules that identify the valid inferences that can be made on provenance graphs.

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 adopt 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; model implementations in XML, RDF or others will be specified in separate documents, in the future.
- We do not specify protocols to store such provenance information in provenance repositories.
- We do not specify protocols to query provenance repositories.

2 Basics

2.1 Entities

Our primary is 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 recognised 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 (like a catalyst) of a process taking place.

The Open Provenance Model is based on these three primary entities, which we define now.

Definition 1 (Artifact) *Immutable piece of state, which may have a physical embodiment in an 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.*

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

We introduce a graphical notation and a formal definition for provenance graphs. Specifically, artifacts are represented by circles, and are denoted by elements of the set $Artifact$. Likewise, processes are represented graphically by rectangles and denoted by elements of the set $Process$. Finally, agents are represented by octogons and are elements of the set $Agent$ in the formal notation.

2.2 Dependencies

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

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. 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 remainder. Roles are meaningful only in the context of the process where they are defined.

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 catalyzed by several agents, we also identify their roles as catalysts. We note that the dependency between an agent and a process represents a control relationship,

[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.

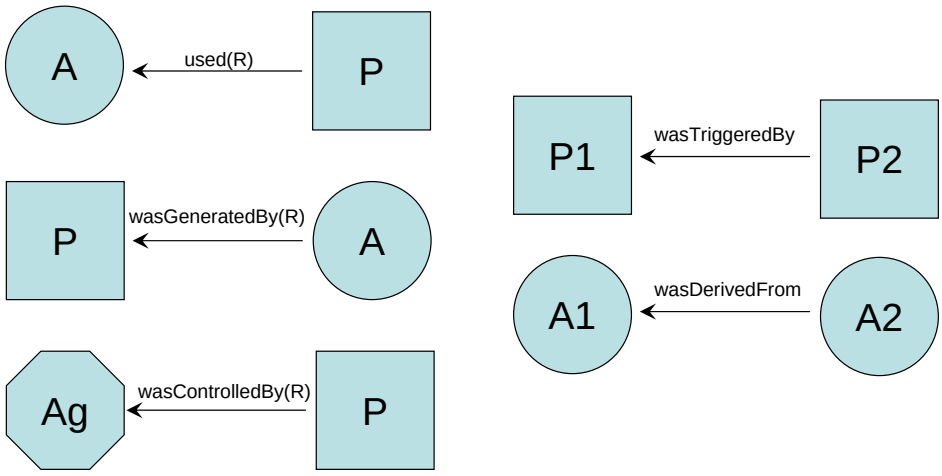

Figure 1: Edges in the Provenance Model

and not a data derivation relationship. It is introduced in the model to easily express how a user (or institution) controlled a process.

It is also recognized that we may not be aware of the process that generated some artifact A_2, but that artifact A_2 was *derived from* another artifact A_1. Likewise, 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. Both edges *wasDerivedFrom* and *wasTriggeredBy* are introduced, because they allow a dataflow or process oriented views of past executions to be adopted, according to the preference of system designers.

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.*

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, would be more appropriate. However, even then, one can distinghish 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 3). 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.

Definition 5 (Artifact Used by a Process) *In a graph, connecting a process to an artifact by a* used *edge is intended to indicate that the process required the availability of the artifact 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 would have required the artifact to be available for the process to be able to start. 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.

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

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

Definition 8 (Artifact Derived from Artifact) *An edge "was derived from" between two artifacts A_1 and A_2 indicates that artifact A_1 may have been used by a process that derived A_2.*

Definition 9 (Process Controlled by Agent) *The assertion of an edge "was controlled by" between a process P and an agent Ag indicates that a start and end of process P was controlled by agent Ag.*

2.3 Roles

A role is an annotation on *used*, *wasGeneratedBy* and *wasControlledBy*.

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 (generate) more than one artifact. Each *used* (*wasGeneratedBy*) relation can be distinguished by a unique role with respect to that process. For example, a process may use several files, reading parameters from one, and reading data from another. The used relations would be labeled with distinct roles.

2. An artifact might be used by more than one process, possibly for different purposes. In this case, the *used* relations can be distinguished or said to be the same by the roles associated with the *used* relations. 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 *wasControlledBy* 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 control function, which would be distinguished by roles associated with the *wasControlledBy* 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").

A role has meaning only within the context of a given process (agent). For a given process, each *used*, *wasGeneratedBy* or *wasControlledBy* relation has a role specific to the process, though the roles may have no meaning outside that process. In general, for a given process (agent) with several arcs, each role should be distinct for that process. However, it is possible, though not recommended, for roles to be the same within a context. 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.

The role is recommended but may be unspecified when not known. It is recommended to give roles whenever possible. For interoperability, communities should define standard sets of roles with agreed meanings. In addition, a reserved value will be defined for "undefined", which should be used when the role is not known or omitted.

2.4 Examples

An example illustrating all the concepts and a few of the causal dependencies is displayed in Figure 2. This provenance graph expresses that John baked a cake with ingredients butter, eggs, sugar and flour.

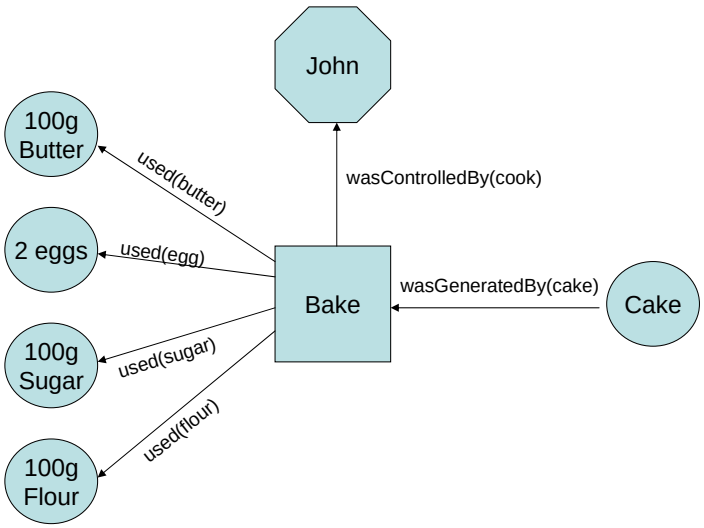

Figure 2: Victoria Sponge Cake Provenance

A computational example is displayed in Figure 3. The final data product is a scientific-grade mosaic of the sky, which was produced by a process that used scientific images in FITS format (such as the Sloan Digital Sky Survey data set) and a parameter indicating the size of the mosaic to be produced. The process was caused by the Pegasus/Condor Dagman agent.

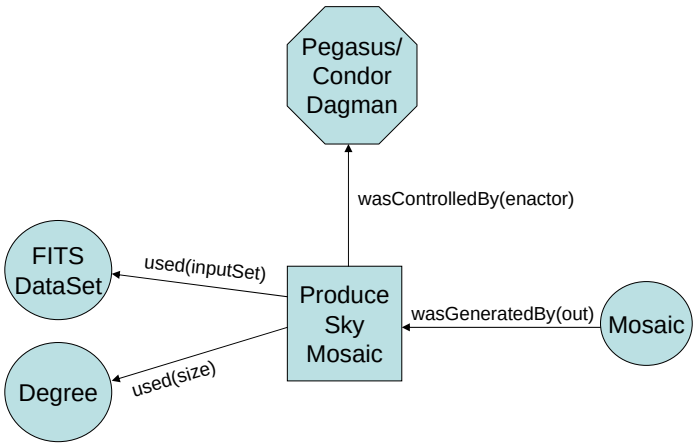

Figure 3: Montage Provenance

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 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 taken place; 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 use such a property to derive transitive closures of causality relations in Section 6. We summarise the properties in the two following definitions.

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.

3 Alternate Descriptions

Figure 4 shows two examples of provenance graphs describing what led the pair (3,7) to being as it is. According to the left hand graph, the pair was generated by a process that added one to all constituents of the pair (2,6). According to the right hand graph, the derivation process of (3,7) required the pair to be created from values 3 and 7, respectively obtained by adding one to 2 and 6, themselves being the data product of splitting the original pair (2,6).

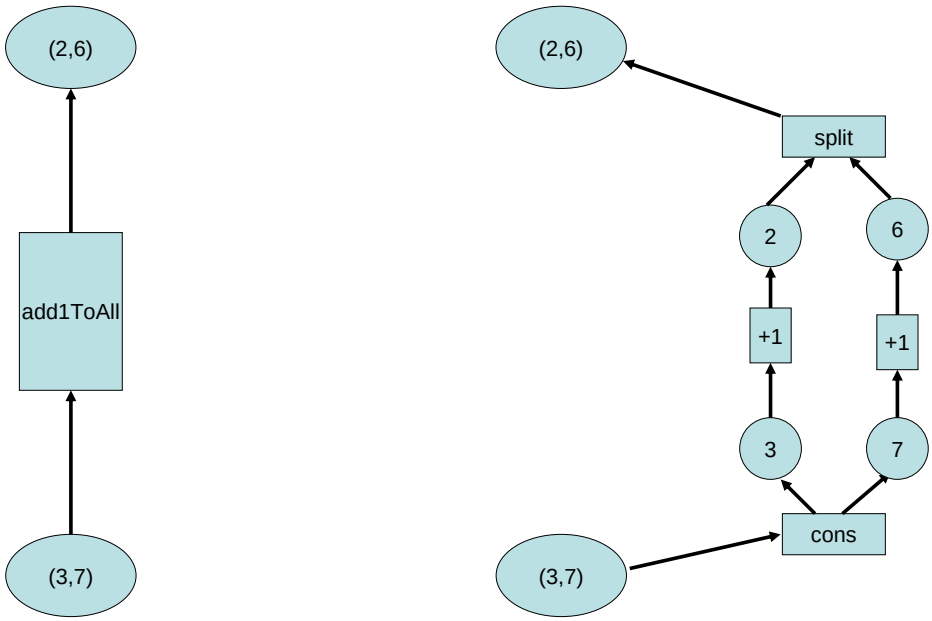

Figure 4: Examples Provenance Graph

Assuming these two graphs refer to the same pairs (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 darker (green) part belonged to the left graph of Figure 4, whereas the lighter (orange) 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 *accounts* of the same past execution, offering different levels of explanation for such execution. Such subgraphs are said to be *alternate accounts*.

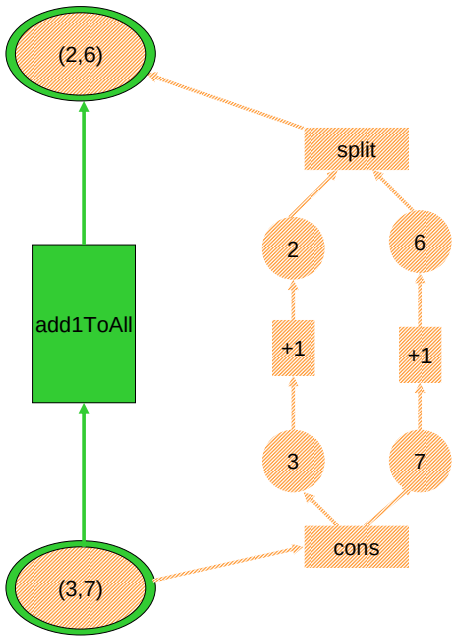

Figure 5: Example of Alternate Accounts in a Provenance Graph

Observing Figure 5, it becomes crucial to contrast the edges originating from artifact (3,7) with those originating from process cons. Indeed, the *used* edges out of the process cons mean that *both* artifacts 3 and 7 were required for the process cons to take place. On the contrary, since the edges out of artifact (3,7) are colored differently, they indicate that alternate explanations exist for the process that led to such artifact being as it is. Using the analogy of AND/OR graphs, a process with *used* edges corresponds to an AND-node, whereas an artifact with *wasGeneratedBy* edges from different accounts represent an OR-node.

While alternate 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).

4 Provenance Graph Definition

The open provenance model is defined according to the following rules, which we formalise in Section 5.

1. *Accounts* are entities that we assume can can be compared.

2. *Artifacts* are identified by unique identifiers. Two artifacts are equal if they have the same identifier. Artifacts can optionally belong to accounts: account membership is declared by listing the accounts an artifact belongs to.

3. *Processes* are identified by unique identifiers. Two processes are equal if they have the same identifier. Processes can optionally belong to accounts: account membership is declared by listing the accounts a process belongs to.

4. *Agents* are identified by unique identifiers. Two agents are equal if they have the same identifier. Agents can optionally belong to accounts: account membership is declared by listing the accounts an agent belongs to.

5. *Edges* are identified by their source, destination, and role (for those that include a role). The source and destination consist of identifiers for artifacts, processes, or agents, according to Figure 1. Two edges are equal if they have the same source, the same destination, and the same role. Edges can also optionally belong to accounts: account membership is defined by listing the accounts an edge belongs to.

6. Roles are mandatory in edges *used*, *wasGeneratedBy* and *wasControlledBy*. The meaning of a role is defined by the semantics of the process they relate to. Role semantics is beyond the scope of OPM.

7. 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 souce and destination of.

8. An *OPM graph* is a set of artifacts, processes, agents, edges, and accounts, as specified above. OPM graphs may be disconnected. 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, 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. Rules below refine the OPM graph concept.

9. A view of an OPM graph according to *one* account, referred to as *account view*, is the set of elements whose effective account membership (for artifacts, processes, and agents) and account membership (for edges) contain the account.

10. While cycles can be expressed in the syntax of OPM, a *legal account view* is defined as an acyclic account view, which contains at most one $wasGeneratedBy$ 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.

11. Hence, a *legal OPM graph* is one for which all account views are legal.

12. Legal account views are OPM graphs. The union of two legal account views is an OPM graph (it is not a legal view since it may contain cycles). The intersection of two legal account views is a legal account view.

13. Two account views can be declared to be *alternate* to express the fact that represent different descriptions of an execution.

14. A declaration that two views are alternate is *legal* if the views have some artifact, process or agent in common.

15. A *provenance graph* is a legal OPM graph where alternate views are legal.

16. *Edges* can optionally be annotated with time information. This aspect will be discussed in Section 7.

17. A provenance graph does not need to contain time annotations.

Having defined the concept of a provenance graph, we now study its formal specification.

5 Timeless Formal Model

Figure 6 provides a set-theoretic definition [7] of the open provenance model, based on the concepts introduced so far. The model of causality we propose is timeless since time precedence does not imply causality: if a process P_1 occurs before a process P_2, in general, we cannot infer that P_1 caused P_2 to happen. However, the converse implication holds assuming time is measured according to a single clock.

Even though the provenance model is timeless, we recognize the importance of time, since time is easily observable by computer systems or users. Hence, in Section 7, we examine how the causality graph can be annotated with time. We will also specify constraints that one would expect time annotations to satisfy (in terms of monotonicity with respect to time) in sound causality graphs.

We assume the existence of a few primitive sets: identifiers for processes, artifacts and agents, roles, and accounts. A given serialization will standardize on these sets, and provide concrete representations for them.

In the model, processes, artifacts and agents are identified by their IDs, and are associated with zero or more accounts — noted $\mathbb{P}(Account)$, the powerset notation. In the set-theoretic notation, identifiers map to the corresponding account membership.

In other words, with a database perspective, elements of $ProcessId$, $ArtifactId$ and $AgentId$ are keys to processes, artifacts and agents, respectively.

The five causality edges can be easily specified by sets *Used*, *WasGeneratedBy*, *WasTriggeredBy*, *WasDerivedFrom*, and *WasControlledBy* making use of identifiers for artifacts, processes or agents, roles, and the associated accounts.

Finally, an OPM graph needs to identify explicitly which accounts are alternate. For this, we use a set *Alternate* enumerating pairs of alternate accounts.

$$
\begin{array}{rcll}
ProcessId & : & primitive\ set & \text{(Process Identifiers)} \\
ArtifactId & : & primitive\ set & \text{(Artifact Identifiers)} \\
AgentId & : & primitive\ set & \text{(Agent Identifiers)} \\
Role & : & primitive\ set & \text{(Roles)} \\
Account & : & primitive\ set & \text{(Accounts)} \\
Process & = & ProcessId \rightarrow \mathbb{P}(Account) & \\
Artifact & = & ArtifactId \rightarrow \mathbb{P}(Account) & \\
Agent & = & AgentId \rightarrow \mathbb{P}(Account) & \\
Used & = & ProcessId \times Role \times ArtifactId \times \mathbb{P}(Account) & \\
WasGeneratedBy & = & ArtifactId \times Role \times ProcessId \times \mathbb{P}(Account) & \\
WasTriggeredBy & = & ProcessId \times ProcessId \times \mathbb{P}(Account) & \\
WasDerivedFrom & = & ArtifactId \times ArtifactId \times \mathbb{P}(Account) & \\
WasControlledBy & = & ProcessId \times Role \times AgentId \times \mathbb{P}(Account) & \\
Alternate & = & Account \times Account & \\
OPMGraph & = & Artifact \times Process & \\
 & & \times Agent \times \mathbb{P}(Used) & \\
 & & \times \mathbb{P}(WasGeneratedBy) \times \mathbb{P}(WasTriggeredBy) & \\
 & & \times \mathbb{P}(WasDerivedFrom) \times \mathbb{P}(WasControlledBy) & \\
 & & \times \mathbb{P}(Alternate) &
\end{array}
$$

Figure 6: Timeless Causality Graph Data Model

The model of Figure 6 specifies all the necessary building blocks for creating OPM graphs. We now revisit the definition provided by Section 4, re-examining each item, and explaining it in terms of the formal model.

1. Accounts are elements of the set *Account*.

2. Artifacts have identifiers belonging to the set $ArtifactId$. For a given set of Artifacts A, and for an artifact id a, account membership is $A(a)$.

3. Processes have identifiers belonging to the set $ProcessId$. For a given set of Processes P, and for a process id p, account memberhsip is $P(p)$.

4. Agents have identifiers belonging to the set $AgentId$. For a given set of agents AG, and for an agen id ag, account memberhsip is $AG(ag)$.

5. For any *used* edges $u_1 = \langle p_1, a_1, r_1, acc_1 \rangle \in Used$ and $u_2 = \langle p_2, a_2, r_2, acc_2 \rangle \in Used$, $u_1 = u_2$ if $p_1 = p_2$, $a_1 = a_2$, $r_1 = r_2$. Likewise for the other edges.

6. The model does not place any constraints on roles, beyond their membership to the set *Role*.

7. We introduce a convenience function *accountOf* operating on artifact ids, process ids, agent ids. For a given OPM graph $\langle A, P, AG, U, G, T, D, C, AL\rangle$, where $A \subseteq Artifact, P \subseteq Process, AG \subseteq Agent$, and $U \subseteq Used, G \subseteq WasGeneratedBy, T \subseteq WasTriggeredBy, D \subseteq WasDerivedFrom, C \subseteq WasControlledBy, AL \subseteq Alternate$

$$\begin{aligned}
accountOf(p) &= P(p) \\
accountOf(a) &= A(a) \\
accountOf(ag) &= AG(ag) \\
accountOf(\langle p, r, a, acc\rangle) &= acc \\
accountOf(\langle a, r, p, acc\rangle) &= acc \\
accountOf(\langle p_1, p_2, acc\rangle) &= acc \\
accountOf(\langle a_1, a_2, acc\rangle) &= acc \\
accountOf(\langle p, r, ag, acc\rangle) &= acc
\end{aligned}$$

We then introduce *effectiveAccountOf*:

$$\begin{aligned}
effectiveAccountOf(p) = \; & accountOf(p) \\
& \cup_{i,j,k} accountOf(\langle p, r_i, a_j, acc_k\rangle \in U) \\
& \cup_{i,j,k} accountOf(\langle a_i, r_j, p, acc_k\rangle \in G) \\
& \cup_{i,j} accountOf(\langle p, p_i, acc_j\rangle \in T) \\
& \cup_{i,j} accountOf(\langle p_i, p, acc_j\rangle \in T) \\
& \cup_{i,j,k} accountOf(\langle p, r_i, ag_j, acc_k\rangle \in C)
\end{aligned}$$

(It is defined similarly for artifacts and agents.)

8. No topological restriction is placed on OPM graphs. For instance, $\langle p, r_1, a, \emptyset\rangle \in U$ and $\langle a, r_2, p, \emptyset\rangle \in G$ are two acceptable edges of an OPM graph, which would create a circularity.

   If $gr_1, gr_2 \in OPMGraph$, then

$$gr_1 \cup gr_2 \in OPMGraph$$

   and

$$gr_1 \cap gr_2 \in OPMGraph.$$

9. For an OPMGraph $gr = \langle A, P, AG, U, G, T, D, C, AL\rangle$, for an account α, $view(\alpha, gr)$ is $\langle A_\alpha, P_\alpha, AG_\alpha, U_\alpha, G_\alpha, T_\alpha, D_\alpha, C_\alpha, AL\rangle$, where:

$$A_\alpha \subseteq A \quad \text{with} \quad A_\alpha = \{(a, acc) \in A \text{ such that } \alpha \in effectiveAccountOf(a)\}$$

$$
\begin{aligned}
P_\alpha \subseteq P \quad &\text{with} \quad P_\alpha = \{(p, acc) \in P \text{ such that } \alpha \in \mathit{effectiveAccountOf}(p)\} \\
AG_\alpha \subseteq AG \quad &\text{with} \quad AG_\alpha = \{(ag, acc) \in AG \text{ such that } \alpha \in \mathit{effectiveAccountOf}(ag)\} \\
U_\alpha \subseteq U \quad &\text{with} \quad U_\alpha = \{\langle p, r, a, acc\rangle \ \in U \text{ such that } \alpha \in acc\} \\
G_\alpha \subseteq G \quad &\text{with} \quad G_\alpha = \{\langle a, r, p, acc\rangle \ \in G \text{ such that } \alpha \in acc\} \\
T_\alpha \subseteq T \quad &\text{with} \quad T_\alpha = \{\langle p_1, p_2, acc\rangle \in T \text{ such that } \alpha \in acc\} \\
D_\alpha \subseteq D \quad &\text{with} \quad D_\alpha = \{\langle a_1, a_2, acc\rangle \in D \text{ such that } \alpha \in acc\} \\
C_\alpha \subseteq C \quad &\text{with} \quad C_\alpha = \{\langle p, ag, acc\rangle \in C \text{ such that } \alpha \in acc\}
\end{aligned}
$$

10. A legal account view $gr = \langle A, P, AG, U, G, T, D, C, AL\rangle$ is such that there is no cycle in U, G, T, D and if $\langle a_1, r_1, p_1, acc_1\rangle \in G$ and $\langle a_1, r_2, p_2, acc_2\rangle \in G$, then $\langle a_1, r_1, p_1, acc_1\rangle = \langle a_1, r_2, p_2, acc_3\rangle$ (see item 5).

11. Two accounts α_1, α_2 are declared to be alternate in an OPMgraph $gr = \langle A, P, AG, U, G, T, D, C, AL\rangle$, if $\langle \alpha_1, \alpha_2\rangle \in AL$ or $\langle \alpha_2, \alpha_1\rangle \in AL$. Hence, the alternate relationship is symmetric.

12. Two accounts α_1, α_2 are declared to be legal alternate in an OPMgraph if they are alternate and if their respective account views $\langle A_1, P_1, AG_1, U_1, G_1, T_1, D_1, C_1, AL_1\rangle$ and $\langle A_2, P_2, AG_2, U_2, G_2, T_2, D_2, C_2, AL_2\rangle$ are such that

$$
\begin{aligned}
& \mathit{Domain}(A_1) \cap \mathit{Domain}(A_2) \neq \emptyset \\
\mathit{or} \quad & \mathit{Domain}(P_1) \cap \mathit{Domain}(P_2) \neq \emptyset \\
\mathit{or} \quad & \mathit{Domain}(AG_1) \cap \mathit{Domain}(AG_2) \neq \emptyset.
\end{aligned}
$$

6 Inferences

The Open Provenance Model has defined the notion of *OPM graph* based on a set of syntactic rules and the notion of *Provenance Graph* adding a set of 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 summarised by means of transitive closure that we describe in this section.

6.1 One Step Inferences

In Section 2, we have introduced the two causal dependencies *wasTriggeredBy* and *wasDerivedFrom* acting as abbreviation for causal dependencies *used* and *wasGeneratedBy*. Figure 7 shows their exact meaning.

Figures 8 and 9 formalize Figure 7 by introducing rules for each inference that can be performed in the Open Provenance Model. A rule consists of two expressions separated

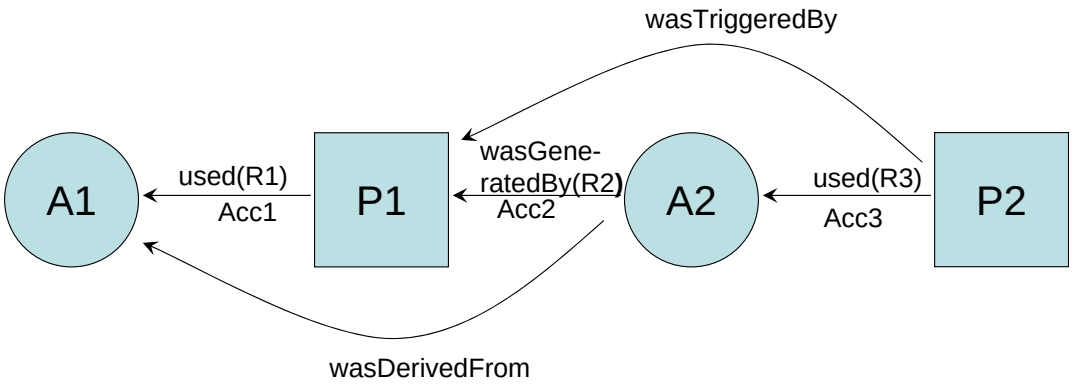

Figure 7: One Step Inference in the Provenance Model

by a horizontal line. The expression above the line is a hypothesis, whereas the expression below the line is a conclusion that can be inferred from the hypothesis.

In Equation (1), a *wasTriggeredBy* edge is inferred from the existence of a *used* and *wasGeneratedBy* edges, as per described in Figure 7. We note that the inferred *wasTriggeredBy* edge relies on both accounts acc_2 and acc_3, hence, it is given $acc_2 \cup acc_3$ as account.

$$\frac{\langle p_2, r_3, a_2, acc_3\rangle \in \mathit{Used} \ \wedge \ \langle a_2, r_2, p_1, acc_2\rangle \in \mathit{WasGeneratedBy}}{\langle p_2, p_1, acc_2 \cup acc_3\rangle \in \mathit{WasTriggeredBy}} \tag{1}$$

$$\frac{\langle p_2, p_1, acc\rangle \in \mathit{WasTriggeredBy}}{\begin{array}{ll}\exists a_2, r_2, r_3, acc_2, acc_3, & \langle p_2, r_3, a_2, acc_3\rangle \in \mathit{Used} \\ & \wedge \langle a_2, r_2, p_1, acc_2\rangle \in \mathit{WasGeneratedBy} \\ & \wedge \ acc_2 \cup acc_3 = acc\end{array}} \tag{2}$$

Figure 8: One Step Inference Rules (1)

Equation (2) is the reverse of Equation (1): it allows us to establish that the edge $\mathit{wasTriggeredBy}(p_2, p_1, acc)$ is hiding the existence of some artifact a_2, used by p_2 and generated by p_1. The inferred edges *used* and *wasGeneratedBy* were asserted in the context of some account acc_2 and acc_1, whose union is the original account *acc*. We note that Equation (2) allows us to establish the existence of some artifact a_2 (and r_1, r_2, acc_1, acc_2) but it does not tell us what their values are. This is the consequence of using *wasTriggeredBy*, which is a lossy summary of the composition of *used* and *wasGeneratedBy*.

The kind of inferences that can be made about *wasDerivedFrom* is of a different nature. Indeed, without any internal knowledge of P_1 in Figure 7, it is impossible to ascertain there is an actual data dependency between A_1 and A_2. This is why Definition 8 adopts a weaker notion of dependency, acknowledging the presence of process that used A_1 and generated A_2. Hence, Equation (3) states that a *wasDerivedFrom* edge can be derived from the existence of a succession of *wasGeneratedBy* and *used* edges. Equation

(4) is to (2) what *wasDerivedFrom* is to *wasTriggeredBy*.

$$\frac{\langle a_2, r_2, p_1, acc_2\rangle \in \mathit{WasGeneratedBy} \wedge \langle p_1, r_1, a_1, acc_1\rangle \in \mathit{Used}}{\langle a_2, a_1, acc_1 \cup acc_2\rangle \in \mathit{WasDerivedFrom}} \tag{3}$$

$$\frac{\langle a_2, a_1, acc\rangle \in \mathit{WasDerivedFrom}}{\begin{array}{ll}\exists p_1, r_1, r_2, acc_1, acc_2, & \langle a_2, r_2, p_1, acc_2\rangle \in \mathit{WasGeneratedBy} \\ & \wedge\ \langle p_1, r_1, a_1, acc_1\rangle \in \mathit{Used} \\ & \wedge\ acc_1 \cup acc_2 = acc\end{array}} \tag{4}$$

Figure 9: One Step Inference Rules (2)

In rules 1 and 3, the inferred edges have accounts $acc_2 \cup acc_3$ and $acc_1 \cup acc_2$, respectively. Hence, the artifacts and processes connected by these edges will have an effective account membership modified accordingly. We note that rules 1 and 3 effectively creates relationships in the union of multiple account views.

6.2 Transitive Closure

Users want to find out the causes of an artifact, not due to one process, but potentially, due to an unknown number of them.

Hence, for the purpose of expressing queries or expressing inferences about provenance graphs, we introduce *four new relationships*, which are transitive versions of existing relationships, namely Used^*, $\mathit{WasGeneratedBy}^*$, $\mathit{WasDerivedFrom}^*$ and $\mathit{WasTriggeredBy}^*$. Their definitions are displayed in Figure 10. We note that Figure 10 contains definitions (as opposed to inference rules of Figures 8 and 9, which specify which edges can be inferred from which edges). For convenience, we have also introduced a generic causal dependency $\mathit{wasDependentOn}^*$ (see equations (9) to (12)).

Equations (7) and (8) are one of the multiple possible ways of defining edges used^* and $\mathit{wasGeneratedBy}^*$. Other definitions could be expressed and proved equivalent (such as used^* can be derived from a single *used* and $\mathit{wasDerivedFrom}^*$).

$$\langle a_2, a_1, acc\rangle \in \mathit{WasDerivedFrom}^* \quad (5)$$
$$\text{if } a_2 = a_1 \vee \exists a_3,\ \langle a_2, a_3, acc_2\rangle \in \mathit{WasDerivedFrom} \wedge \langle a_3, a_1, acc_1\rangle \in \mathit{WasDerivedFrom}^* \wedge acc = acc_1 \cup acc_2$$

$$\langle p_2, p_1, acc\rangle \in \mathit{WasTriggeredBy}^* \quad (6)$$
$$\text{if } p_2 = p_1 \vee \exists p_3,\ \langle p_2, p_3, acc_2\rangle \in \mathit{WasTriggeredBy} \wedge \langle p_3, p_1, acc_1\rangle \in \mathit{WasTriggeredBy}^* \wedge acc = acc_1 \cup acc_2$$

$$\langle p, a, acc\rangle \in \mathit{Used}^* \quad (7)$$
$$\text{if } \exists p_2, r, acc_1, acc_2,\ \langle p, p_2, acc_2\rangle \in \mathit{WasTriggeredBy}^* \wedge \langle p_2, r, a, acc_1\rangle \in \mathit{Used} \wedge acc = acc_1 \cup acc_2$$

$$\langle A, P, acc\rangle \in \mathit{WasGeneratedBy}^* \quad (8)$$
$$\text{if } \exists p_2, R, acc_1, acc_2,\ \langle A, R, p_2, acc_2\rangle \in \mathit{WasGeneratedBy} \wedge \langle p_2, P, acc_1\rangle \in \mathit{WasTriggeredBy}^* \wedge acc = acc_1 \cup acc_2$$

$$\langle A, P, acc\rangle \in \mathit{WasDependentOn}^* \quad \text{if} \quad \langle A, P, acc\rangle \in \mathit{WasGeneratedBy}^* \quad (9)$$
$$\langle a_1, a_2, acc\rangle \in \mathit{WasDependentOn}^* \quad \text{if} \quad \langle a_1, a_2, acc\rangle \in \mathit{WasDerivedFrom}^* \quad (10)$$
$$\langle p_1, p_2, acc\rangle \in \mathit{WasDependentOn}^* \quad \text{if} \quad \langle p_1, p_2, acc\rangle \in \mathit{WasTriggeredBy}^* \quad (11)$$
$$\langle P, A, acc\rangle \in \mathit{WasDependentOn}^* \quad \text{if} \quad \langle P, A, acc\rangle \in \mathit{Used}^* \quad (12)$$

Figure 10: Transitive Closures

7 Formal Model and Time Annotations

The Open Provenance Model allows for causality graphs to be annotated with time annotations. 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. However, time may have been observed during the course of a process, and we would expect such time information to be compatible with causal dependencies: the time of an effect should be greater than the time of its cause (for a same clock). 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 instant 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 will be 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 instantenous. 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 is not usefully modelled as an instantatenous 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 instanenous beginnings and endings.

In the Open Provenance Model, time information is expected to be obtained by *observing* a clock 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 annotated by 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 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 or no later than time t_2.

In Figure 11, we revisit our formal model, examining where time annotations are permitted. We first introduce a new primitive set *Time*, for which a given serialization will specify a format (such as the standard coordinated universal time, UTC). We then introduce *Observed Time* as a pair of time values (whose set is $OTime$). All time annotations are optional, which we note by $OTime^0$ in the definitions.

Edges involve $OTime$ in their cartesian product. Edges from *WasGeneratedBy* and *Used* can be annotated by an *optional* timestamp, marking the associated artifact was

$$
\begin{array}{rcll}
ProcessId & : & primitive\ set & \text{(Process Identifiers)} \\
Artifact Id & : & primitive\ set & \text{(Artifact Identifiers)} \\
AgentId & : & primitive\ set & \text{(Agent Identifiers)} \\
Role & : & primitive\ set & \text{(Roles)} \\
Account & : & primitive\ set & \text{(Accounts)} \\
Time & : & primitive\ set & \text{(Time)} \\
Process & = & ProcessId \rightarrow \mathbb{P}(Account) & \\
Artifact & = & Artifact Id \rightarrow \mathbb{P}(Account) & \\
Agent & = & AgentId \rightarrow \mathbb{P}(Account) & \\
OTime & = & Time \times Time & \text{(Observed Time)} \\
Used & = & ProcessId \times Role \times Artifact Id \times \mathbb{P}(Account) \ \times OTime^0 & \\
WasGeneratedBy & = & Artifact Id \times Role \times ProcessId \times \mathbb{P}(Account) \ \times OTime^0 & \\
WasTriggeredBy & = & ProcessId \times ProcessId \times \mathbb{P}(Account) \ \times OTime^0 & \\
WasDerivedFrom & = & Artifact Id \times Artifact Id \times \mathbb{P}(Account) \ \times OTime^0 & \\
WasControlledBy & = & ProcessId \times Role \times AgentId \times \mathbb{P}(Account) \ \times OTime^0 \times OTime^0 & \\
Alternate & = & Account \times Account & \\
OPMGraph & = & Artifact \times Process & \\
 & & \times Agent \times \mathbb{P}(Used) & \\
 & & \times \mathbb{P}(WasGeneratedBy) \times \mathbb{P}(WasTriggeredBy) & \\
 & & \times \mathbb{P}(WasDerivedFrom) \times \mathbb{P}(WasControlledBy) & \\
 & & \times \mathbb{P}(Alternate) &
\end{array}
$$

Figure 11: Causality Graph Data Model and Time Annotations

known to be generated or used, at a given time (expressed as an observation interval).

For *WasControlledBy*, we allow two *optional* timestamps marking when the process was known to be started or terminated, respectively.

For *WasDerivedFrom*, we also allow one *optional* timestamp. Given Figure 7 and associated inferences, for a given edge $\langle a_1, a_2, acc\rangle \in WasDerivedFrom$, there is an implicit process that generated a_1 and that consumed a_2. The time annotation indicates when the artifact was generated.

Likewise, for *WasTriggeredBy*, we also allow one *optional* timestamp. Given Figure 7 and associated inferences, for a given edge $\langle p_1, p_2, acc\rangle \in WasTriggeredBy$, there is an implicit artifact that was used by p_1 and generated by p_2. The time annotations indicates the time when the artifact was used by p_1.

8 Time Constraints and Inferences

The model of causality in OPM is essential timeless since time precedence does not imply causality: if a process P_1 occurs before a process P_2, in general, we cannot infer that P_1 caused P_2 to happen. However, the converse implication holds assuming time is measured according to a single clock.

We therefore expect time annotations to be consistent with causality. To this end, we extend the definition of legal account view, defined as: an acyclic account view, which contains at most one *wasGeneratedBy* edge per artifact, and in which *causation is time-monotonic*, as displayed in Figure 13, and discussed below.

We remind the reader that all observed times are pairs of instanteous 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:

$$T_1 < T_2 \quad if \quad t_1^m \leq t_1^M < t_2^m \leq t_2^M$$
$$T_1 \leq T_2 \quad if \quad t_1^m \leq t_1^M \leq t_2^m \leq t_2^M$$

According to Figure 12, an artifact must exist before it is being used ($T_1 < T_3$ and $T_4 < T_6$). If an artifact is used by a process, it will actually be used after the start of the process ($T_2 < T_3$). A process generates artifacts before its end ($T_4 < T_5$), and a process starts precedes its generation of artifacts ($T_2 < T_4$) and its end ($T_2 < T_5$).

Equipped with these definitions, Figure 13 formally states the time constraints illustrated by Figure 12.

Equation (13) states that generation of an artifact precedes its use. Equation (14) requires a process to start before it uses artifacts, but after the artifact that caused it was generated; the use of the artifact taking place before the end of the process.

Equation (15) states that generation of an artifact by a process is preceded by the start of the process and takes place before the end of the process.

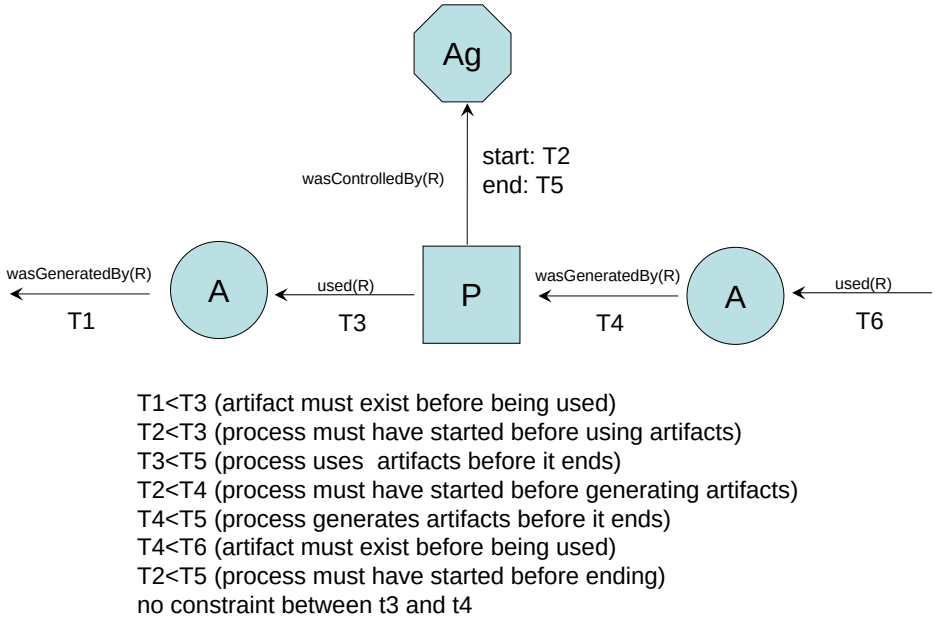

Figure 12: Time Constraints in the Open Provenance Model

$$\frac{used(p_1, r_1, a, acc_1, T_3) \ \wedge \ wasGeneratedBy(a, r_2, p_2, acc_1, T_1)}{T_1 < T_3} \tag{13}$$

$$\frac{used(p, r_1, a, acc_1, T_3) \ \wedge \ wasControlledBy(p, r_3, ag, acc_1, T_2, T_5)}{T_2 < T_3, T_3 < T_5} \tag{14}$$

$$\frac{wasGeneratedBy(a, r_2, p, acc_1, T_4) \ \wedge wasControlledBy(p, r_3, ag, acc_1, T_2, T_5)}{T_2 < T_4, T_4 < T_5} \tag{15}$$

Figure 13: Causation is Time-Monotonic

9 Example of Representation

In this Section, we construct an explicit representation of the model for Figure 4. It appears in Figure 14, where we used the symbols O and G to denote orange and green accounts, respectively.

10 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. In the future, we will define a serialization format for this model. We will also specify protocols by which provenance of artifacts can be determined, and protocols for applications to record descriptions of their execution. We invite teams that have defined their own provenance model to establish whether their representations can be converted into this model and vice-versa.

$$
\begin{aligned}
ProcessID &= \{p_1, p_2, p_3, p_4, p_5\} \\
ArtifactID &= \{a_1, a_2, a_3, a_4, a_5, a_6\} \\
Account &= \{\mathsf{G}, \mathsf{O}\} \\
P \subseteq Process &= \\
\{\ p_1 &\rightarrow \{\mathsf{G}\}, \\
p_2 &\rightarrow \{\mathsf{O}\}, \\
p_3 &\rightarrow \{\mathsf{O}\}, \\
p_4 &\rightarrow \{\mathsf{O}\}, \\
p_5 &\rightarrow \{\mathsf{O}\}\ \} \\
A \subseteq Artifact &= \\
\{\ a_1 &\rightarrow \{\mathsf{G}, \mathsf{O}\}, \qquad //(2, 6) \\
a_2 &\rightarrow \{\mathsf{G}, \mathsf{O}\}, \qquad //(3, 7) \\
a_3 &\rightarrow \{\mathsf{O}\}, \qquad //2 \\
a_4 &\rightarrow \{\mathsf{O}\}, \qquad //6 \\
a_5 &\rightarrow \{\mathsf{O}\}, \qquad //3 \\
a_6 &\rightarrow \{\mathsf{O}\}\ \} \qquad //7 \\
u \subseteq Used &= \\
&\{ \quad used(p_1, in, a_1, \{\mathsf{G}\}), \\
&\quad\ used(p_2, pair, a_1, \{\mathsf{O}\}), \\
&\quad\ used(p_3, in, a_3, \{\mathsf{O}\}), \\
&\quad\ used(p_4, in, a_4, \{\mathsf{O}\}), \\
&\quad\ used(p_5, left, a_5, \{\mathsf{O}\}), \\
&\quad\ used(p_5, right, a_6, \{\mathsf{O}\})\ \} \\
g \subseteq WasGeneratedBy &= \\
&\{ \quad wasGeneratedBy(a_2, out, p_1, \{\mathsf{G}\}) \\
&\quad\ wasGeneratedBy(a_3, left, p_2, \{\mathsf{O}\}), \\
&\quad\ wasGeneratedBy(a_4, right, p_2, \{\mathsf{O}\}), \\
&\quad\ wasGeneratedBy(a_5, out, p_3, \{\mathsf{O}\}), \\
&\quad\ wasGeneratedBy(a_6, out, p_4, \{\mathsf{O}\}), \\
&\quad\ wasGeneratedBy(a_2, pair, p_5, \{\mathsf{O}\})\ \} \\
a \subseteq Alternate &= \\
&\{ \quad alternate(\mathsf{O}, \mathsf{G})\ \}
\end{aligned}
$$

Figure 14: Representation of Figure 4

A Best Practice on the Use of Agents

With the defined notion of account, we now revisit the sky mosaic example. Instead of Figure 3, a different description could encompass the steps the operating system (or the grid) goes through in order to execute a program (as in the PASS and ES3 approaches). Figure 15 illustrates *some* possible causal dependencies for a system-level description. Here, we see an explicit reference to the workflow script used by the enactor.

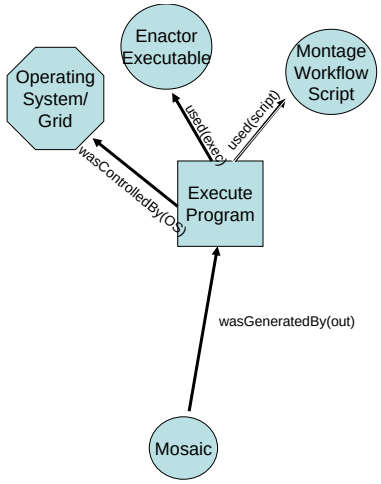

Figure 15: Alternate Montage Provenance

Naturally, both descriptions can coexist in a same provenance graph, using the concept of alternate descriptions, as depicted by Figure 16. While such a description is perfectly acceptable, it fails to tell us that the agent Pegasus/Condor Dagman is this executable, which itself was activated under the control of the operating system (or Grid).

In other circumstances, it is necessary to explain that multiple agents were all controlling a same process, but from different perspective. For the case present, the researcher who controlled the experiment, the enactment engine, and the funding institution are all potential causes of the experiment. We then obtain Figure 17, where we see three processes triggering the production of a mosaic. Further experience will the model will allow us to identify guidelines to promote inter-operability of systems.

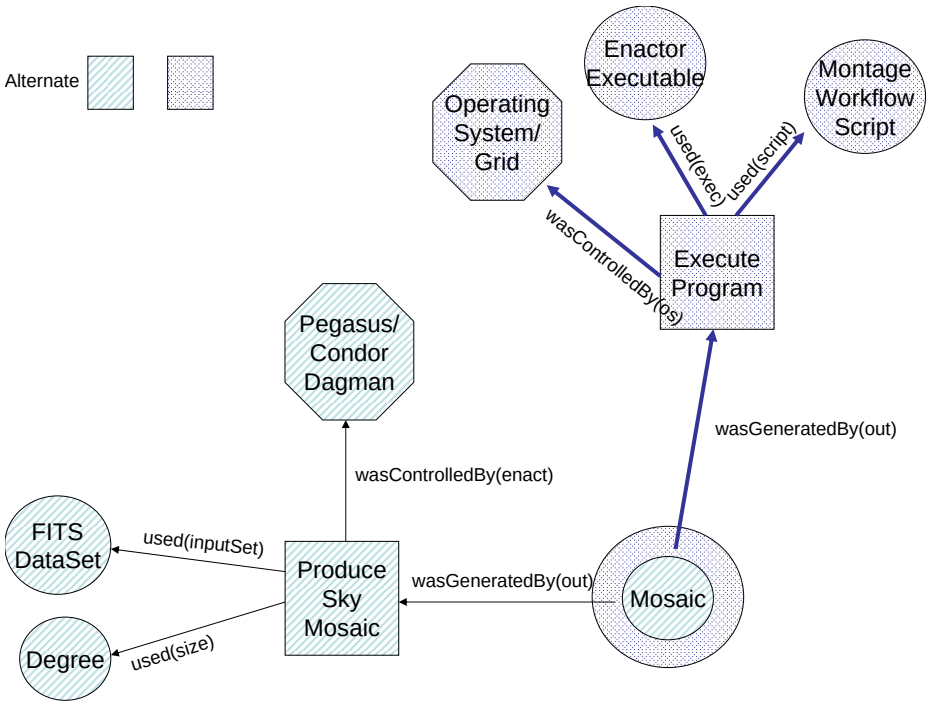

Figure 16: Montage Provenance

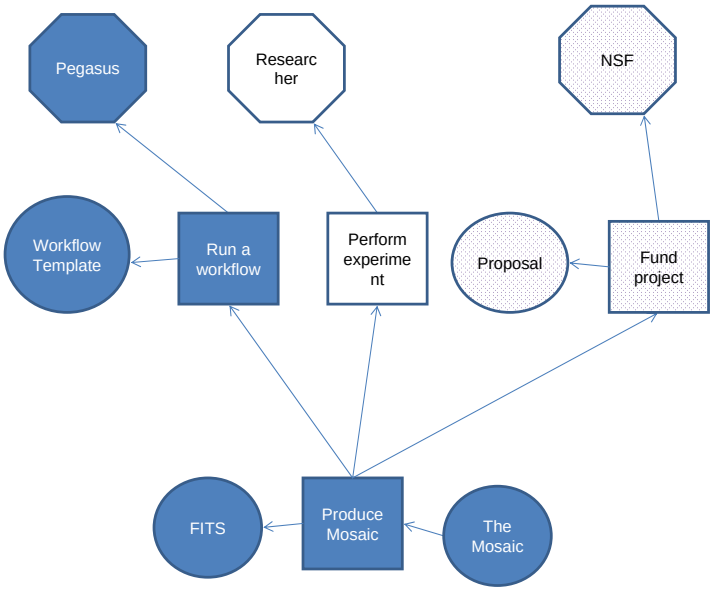

Figure 17: Multiple Agents Controlling a Process

References

[1] Raj Bose, Ian Foster, and Luc Moreau. Report on the International Provenance and Annotation Workshop (IPAW06). *Sigmod Records*, 35(3):51–53, September 2006.

[2] Ewa Deelman and Yolanda Gil (Eds.). Workshop on the challenges of scientific workflows. Technical report, Information Sciences Institute, University of Southern California, May 2006.

[3] PREMIS Working Group. Data dictionary for preservation metadata — final report of the premis working group. Technical report, Preservation Metadata: Implementation Strategies (PREMIS), 2005.

[4] Simon Miles. Technical summary of the second provenance challenge workshop. http://twiki.ipaw.info/bin/view/challenge/SecondWorkshopMinutes, King's College, July 2007.

[5] Luc Moreau and Ian Foster, editors. *Provenance and Annotation of Data — International Provenance and Annotation Workshop, IPAW 2006*, volume 4145 of *Lecture Notes in Computer Science*. Springer-Verlag, May 2006.

[6] Luc Moreau and Bertram Ludaescher, editors. *Special Issue on the First Provenance Challenge*. Wiley, 2007.

[7] David A. Schmidt. *Denotational Semantics. A Methodology for Language Development*. Brown Publishers, 1986.

[8] Second challenge team contributions. http://twiki.ipaw.info/bin/view/Challenge/ParticipatingTeams, June 2007.

[9] Y. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *SIGMOD Record*, 34(3):31–36, September 2005.