A Formal Account of the Open Provenance Model

Natalia Kwasnikowska
Hasselt University and Transnational University of Limburg
Belgium

Luc Moreau
University of Southampton
United Kingdom

Jan Van den Bussche
Hasselt University and Transnational University of Limburg
Belgium

December 2010

Abstract

The Open Provenance Model (OPM) is a community data model for provenance that is designed to facilitate the meaningful interchange of provenance information between systems. Underpinning OPM, is a notion of directed graph, used to represent data products and processes involved in past computations, and dependencies between them; it is complemented by inference rules allowing new dependencies to be derived. The Open Provenance Model was designed from requirements captured in two "Provenance Challenges", and tested during the third: these challenges were international, multi-disciplinary activities aiming to exchange provenance information between multiple systems and query it. The design of OPM was mostly driven by practical and pragmatic considerations. The purpose of this paper is to formalize the theory underpinning this data model. Specifically, this paper proposes a temporal semantics for OPM graphs, defined in terms of a set of ordering constraints between time-points associated with OPM constructs. OPM inferences are characterized with respect to this temporal semantics, and a novel set of patterns is introduced to establish soundness and completeness properties. Building on this novel foundation, the paper proposes new definitions for graph algebraic operations, graph refinement and the notion of account, by which multiple descriptions of a same execution are allowed to co-exist in a same graph. Overall, this paper provides a strong theoretical underpinning to a data model being adopted by a community of users that help its disambiguation and promote inter-operability.

1 Introduction

In the fine arts and in digital libraries, provenance respectively refers to the documented history of an art object, or the documentation of processes in a digital object's life cycle [24]. The "e-science community" [27] also shows a growing

interest in provenance since it is crucial to ensure reproducibility of scientific experiments [7]. At the World Wide Web consortium, the Provenance Incubator [31] has also demonstrated the importance of provenance on the Web; in this context, it defines the provenance of a resource as a *record that describes entities and processes involved in producing and delivering or otherwise influencing that resource.*

Over the years, a series of systems have been developed to track and exploit provenance, in order to address many different requirements [16]. In most modern applications, information flows across multiple systems, implemented using different technologies, and potentially hosted by different institutions; tracking the provenance of data in this context is particular challenging, since it involves understanding flows of information in these different systems [19].

It is not anticipated that a single methodology for tracking provenance could readily be deployed to technologies as varied as database systems, workflow systems and web services stacks. Instead, it has been proposed that a provenance interchange language could be adopted [19], which would allow systems to convert their internal provenance representation, into a provenance lingua franca. In the course of a series of inter-operability Provenance Challenges [21, 29], a data model for exchanging provenance information has been designed, implemented and used in practice. This data model, referred to as the Open Provenance Model (OPM) [20], has already undergone several revisions, using an open-source like governance mechanism to manage changes.

Despite its fairly recent development, OPM is getting significant traction beyond its initial set of designers. OPM is used by data.gov.uk to track provenance of data published by the UK government [33], it inspired the design of a provenance-based policy language [25], and it is adopted by the SHIWA project (shiwa-workflow.eu), to ensure coarse-grained inter-operability of workflow systems. Furthermore, in a quest to understand emerging provenance models, the W3C Incubator Group on Provenance decided to map from their concepts to a single target model, and adopted OPM as its target model. The Incubator Group found that the emerging models for provenance, despite being originated from a wide range of domains, map well to terms and extensibility mechanisms defined in OPM [26].

From the outset, OPM was described in a technology-agnostic manner. The key data structure defined in the Open Provenance Model is an *OPM graph*, a directed graph aimed at representing data and control dependencies of past computations; such graphs can be decorated with time information for specific events associated with OPM constructs. The specification also outlined the kind of inferences that are permitted over such graphs, and some constraints that graph topology and time information must preserve. OPM is informally defined in the OPM specification [20], which we refer to as the *OPM reference specification.*

A criticism of the Open Provenance Model is that it does not provide a formal semantics [4]: the lack of unambiguous concept definition potentially hinders the development of mappings [26] to other provenance languages, and ultimately can challenge inter-operability of systems.

The aim of this paper is to address this shortcoming by providing a formal semantics for the Open Provenance Model. First, we formalize the notion of an OPM graph, and equip it with a temporal theory. This kind of semantics maps an OPM graph to a set of ordering constraints between time-points. Such

temporal theory allows for new inferences to be made, by applying a transitive closure on the ordering constraints; the set of inferred constraints, which we refer to as logical consequences, offers a purely semantic definition of an OPM graph. Second, we axiomatize the inference rules for OPM graphs as outlined in the OPM reference specification [20], and characterize them with respect to the semantic definition. Specifically, we characterize logical consequences in OPM graphs, by establishing soundness and completeness properties for a novel set of graph patterns. Leveraging the formalization of OPM graphs, we finally specify useful graph algebraic operations, and define a notion of refinement, which allows different OPM accounts[1] to be related.

A key motivation of this paper has been to provide a semantics for an *existing* informal data model. In doing so, we have refrained from introducing mathematical artifacts, which could have simplified some definitions, but would not have corresponded to OPM as defined by a practitioner's community. We have also tackled the complete OPM data model (except the notion of agency), and have not restricted us to a smaller, more manageable subset. As part of this formalization, and to be able to characterize inferences properly, we have introduced two modifications to the data model, which we regard as minor, but bring substantial benefit; we discuss these in detail in the paper. For this paper to remain of interest to a broad audience, we grouped a set of notes in Section 11, covering technical details of the OPM reference specification, and their relationship with the semantics introduced in this paper. When relevant, we refer to these notes in the paper.

We would like to stress that a temporal approach is only one possible approach to give a semantics to OPM graphs. It is an interesting topic for future research to explore alternative approaches, e.g., as suggested by Cheney [2].

This paper is organized as follows. As we aim for this paper to be self-contained, we begin with a brief, informal and intuitive overview of OPM, by means of a concrete example in Section 2. Section 3 formally defines OPM graphs and their temporal interpretation. In Section 4, the notion of OPM inference, which allows new edges to be inferred, is defined and characterized with respect to the temporal semantics. Given that OPM graphs are meant to be exchanged and manipulated to address provenance use cases, we formalize common operations over OPM graphs in Section 5. From the outset, it was envisaged that relations between OPM graphs, such as refinement, would be of value to reasoners; however, no precise definition of refinement has been proposed so far. This problem is tackled in Section 6, where a purely semantic definition of refinement is proposed, based on the temporal semantics. Finally, the notion of account is formalized in Section 7, before related work is discussed in Section 8.

2 OPM Overview

The purpose of this section is to overview the Open Provenance Model [20] and provide intuition about its key components. The rest of the paper is then dedicated to its formalization. Of course, the present brief overview cannot

[1] We shall see that an OPM account is a mechanism that allows multiple descriptions of past execution to co-exist in a single OPM graph.

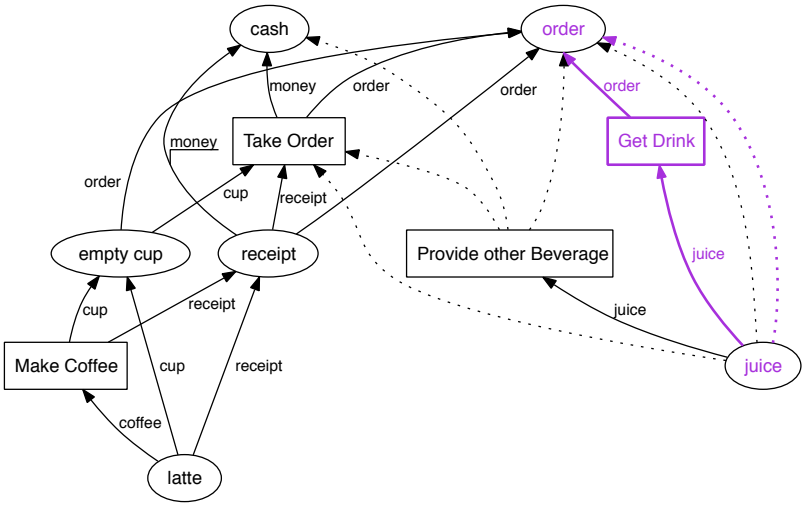

Figure 1: OPM graph for coffee shop order. The dotted edges are imprecise.

replace the detailed presentation of OPM, as given in the OPM reference specification [20], nor can it encompass the precise characterization, found in the rest of this paper. Our intuitive overview of OPM draws on the following scenario:

> Alice and her young son Bob ordered a latte and a fruit juice in a coffee shop. Bob, who is a young child, did not observe the activities involved in processing their order, and only focused on his own drink. Hence, Bob's version of events is that an order was submitted and resulted in his juice being delivered.
>
> Alice, who could observe the activities behind the counter, identified three different processes. The cashier took the order and associated payment. As soon as the order was taken, the cashier put an empty cup on a tray next to the coffee machine; once payment was taken, the cashier added a till receipt to the same tray. The coffee machine operator picked up the cup, and filled it with the requested coffee, as per receipt, and handed the tray over to Alice. A third person behind the counter served other drinks, on request from the cashier. Alice was unable to ascertain how information was communicated (e.g., the request was stated by cashier, or order read from receipt); what is definite from Alice's viewpoint is the juice was also delivered with the tray.

The OPM data model consists of a directed graph, whose nodes are artifacts and processes, and edges are dependencies between them. Artifacts in this scenario consist of an "order", some "cash", an "empty cup", a "receipt", a "juice" and a "latte". According to Bob, there is a single process: "Get Drink". Alice's version of events is more detailed and involves three processes: "Take Order", "Make Coffee", and "Provide other Beverages".

OPM edges are directional: an edge source represents an *effect* and an edge destination a *cause*. There exist four types of edges according to the types of

Table 1: OPM processes and artifacts for coffee shop order

node type	node	label
process	p_1	Take Order
process	p_2	Make Coffee
process	p_3	Provide other Beverage
process	p_4	Get Drink
artifact	a_1	order
artifact	a_2	cash
artifact	a_3	empty cup
artifact	a_4	receipt
artifact	a_5	latte
artifact	a_6	juice

effect and cause. A *used*-edge is between a process and an artifact; a *generated-by* edge is between an artifact and a process; a *derived-from* edge is between two artifacts; and an *informed-by* edge is between two processes.

Moreover, the first three types of edges are further categorized into *precise* and *imprecise* versions. Precise edges are labeled with *roles*, which indicate the role in which artifacts are used and generated; roles are comparable to parameter positions in a procedure. Imprecise edges are used when precise information about what happened is not important or not available.

Nodes are listed in Table 1, and edges in Table 2. Nodes and edges are displayed graphically in Figure 1. The "Take Order" process used two artifacts, "order" and "cash", and generated the "empty cup". The latter was used by the process "Make Coffee", to generate a "latte".

We note that the empty cup was put on the tray, before payment was taken. So there is no edge from the "empty cup" to "cash". On the other hand, the receipt was put on the tray after cash was received, which explains the presence of an edge from "receipt" to "cash".

Furthermore, some OPM edges can be decorated with time information (not represented explicitly in the figure and table). Specifically, the time associated with a precise used-edge denotes the time at which an artifact was used by a process; likewise, the time associated with a precise generated-by edge denotes the time at which an artifact was generated by a process. Moreover, processes may be given a beginning and an ending time.

OPM specifies some constraints between such time information and the graph structure. For instance, let t be the time associated with (p_1, money, a_2). Time t represents the time at which p_1 ("Take Order") used artifact a_2 ("cash"), with role "money". Time t is required to precede the ending of p_1, and to follow the beginning of p_1. Note also that p_1 may well be finished before artifact a_5 ("latte") was actually produced. This paper formalizes all constraints identified by OPM.

We note that Bob's version of events is represented in the same graph as Alice's version. In the graphical representation of Figure 1, these descriptions are distinguished by color (black for Alice's version, and violet for Bob's; artifacts "order" and "juice" belong to both versions.). To support multiple descriptions

Table 2: OPM edges for coffee shop order

edge type	source /effect	destination /cause	asserted edges for Figure 1
precise generated-by	artifact	process	(a_3, cup, p_1), $(a_4, \text{receipt}, p_1)$, $(a_5, \text{coffee}, p_2)$,$(a_6, \text{juice}, p_3)$ (a_6, juice, p_4)
precise used	process	artifact	(p_1, money, a_2), (p_1, order, a_1), $(p_2, \text{receipt}, a_4)$, (p_2, cup, a_3), (p_4, order, a_1)
precise derived-from	artifact	artifact	(a_3, order, a_1), (a_4, order, a_1), (a_4, money, a_2), (a_5, cup, a_3), $(a_5, \text{receipt}, a_4)$
generated-by	artifact	process	(a_6, p_1)
used	process	artifact	(p_3, a_1), (p_3, a_2)
derived-from	artifact	artifact	(a_6, a_1)
informed-by	process	process	(p_3, p_1)

of an execution, OPM introduces a notion of account. An account is a subgraph, which is also an OPM graph, as represented by colors in Figure 1.

3 OPM graphs and their temporal semantics

The OPM reference specification [20] defines the proposed data model only informally. The purpose of this section is to provide a temporal semantics to OPM graphs, the data structure introduced in the reference specification.

3.1 OPM graphs

We begin by formally defining OPM graphs. Our definition is slightly more detailed in distinguishing between precise and imprecise edges. In the OPM reference specification, OPM graphs have different accounts, but we defer the treatment of accounts to Section 7.

In our formalization, OPM graphs consist of nodes and edges. Nodes can be of two types: artifacts and processes (Note 11.1). There are four types of edges: generated-by, used, derived-from, and informed-by, depending on the type of their source and destination (Note 11.2).

The edges are further categorized into *precise* and *imprecise* edges. Precise edges are syntactically marked by the presence of roles to characterize the nature of the relationship between the source and destination of the edge. Intuitively, OPM roles are to used-edges, what parameter positions are to procedures in programming languages; likewise, roles in a generated-by edge identify the nature of an output generated by a process; finally, roles in a derived-from edge characterize the precise usage of an artifact by a process. By contrast, imprecise edges do not have roles; they represent incomplete information (Note 11.3).

This paper provides a semantic interpretation of precise and imprecise edges.

Definition 3.1 (OPM graph)**.** An OPM graph is a structure

$$(\mathit{Art}, \mathit{Proc}, \mathit{Roles}, \mathit{GeneratedBy}!, \mathit{Used}!, \mathit{DerivedFrom}!, \\ \mathit{GeneratedBy}, \mathit{Used}, \mathit{DerivedFrom}, \mathit{InformedBy})$$

where

- Art and Proc are two disjoint finite sets of elements called *artifacts* and *processes*, respectively;
- Roles is a finite set of elements called *roles*;
- $\mathit{GeneratedBy}! \subseteq \mathit{Art} \times \mathit{Roles} \times \mathit{Proc}$;
- $\mathit{Used}! \subseteq \mathit{Proc} \times \mathit{Roles} \times \mathit{Art}$;
- $\mathit{DerivedFrom}! \subseteq \mathit{Art} \times \mathit{Roles} \times \mathit{Art}$;
- $\mathit{GeneratedBy} \subseteq \mathit{Art} \times \mathit{Proc}$;
- $\mathit{Used} \subseteq \mathit{Proc} \times \mathit{Art}$;
- $\mathit{DerivedFrom} \subseteq \mathit{Art} \times \mathit{Art}$;
- $\mathit{InformedBy} \subseteq \mathit{Proc} \times \mathit{Proc}$.

Artifacts and processes are collectively referred to as the *nodes* of an OPM graph. The elements of $\mathit{GeneratedBy}! \cup \mathit{Used}! \cup \mathit{DerivedFrom}!$ are called *precise edges*, and the elements of $\mathit{GeneratedBy} \cup \mathit{Used} \cup \mathit{DerivedFrom} \cup \mathit{InformedBy}$ are called *imprecise edges*; all together they are called edges (Note 11.4). Precise edges are of the form (x, r, y) and are additionally denoted as $x \xrightarrow{r} y$, or, when it is not important to know r, as $x \xrightarrow{!} y$. Imprecise edges (x, y) are denoted simply as $x \to y$. When the distinction between precise and imprecise derived-from edges is of no consequence, we use the following set to refer to all derived-from edges of an OPM graph:

$$\mathit{DerivedEdges} = \mathit{DerivedFrom} \cup \{(A, B) \mid (A, r, B) \in \mathit{DerivedFrom}!\} \ .$$

This definition of graph allows for multiple precise used-edges between a same process-artifact pair with multiple roles. They would indicate that during its lifetime a process used a same artifact several times, with different roles.

The OPM reference specification defines a notion of legal graph as a directed graph without cycles in the derived-from edges, in which each artifact is generated by at most one process. For now, we relax the constraint on the derived-from edges, which we revisit in Section 5.2, and refine such a notion of legality in the context of our formalization (Note 11.5).

Definition 3.2 (Legal OPM graph)**.** An OPM graph is called *legal* if

- for each artifact A there is at most one process P with a precise generated-by edge $A \xrightarrow{!} P$; and
- for each precise derived-from edge $A \xrightarrow{r} B$ there is a process P with precise edges $A \xrightarrow{!} P$ and $P \xrightarrow{r} B$, for the same role r.

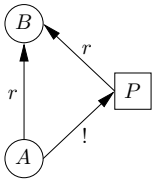

Figure 2: A use–generate-derive triangle (A, B, P, r).

A configuration (A, B, P, r) as above, with edges $A \xrightarrow{r} B$, $A \xrightarrow{!} P$, and $P \xrightarrow{r} B$, is called a *use–generate–derive triangle*, or simply *triangle* for short (see Figure 2). To denote that a use–generate–derive triangle (A, B, P, r) occurs in some given OPM graph G, we use the notation $G \triangle (A, B, P, r)$.

A use–generate–derive triangle offers an insight into the inner workings of a process P, since not only does it state that B was used by P in role r and A generated according to a role, but also does it state that B had a direct influence on A, because it was used in this precise role r.[2] A typical usage of a use–generate–derive triangle is for a division process, illustrated in the following example.

Example 3.3. Let / be a division process, 8 and 4 be its inputs (in respective capacity of dividend and divisor), and the quotient 2 be its output. So, edges are as follows:

edge type	source	destination	
precise generated-by	artifact	process	$(2, \mathit{quotient}, /)$
precise used	process	artifact	$(/, \mathit{dividend}, 8)$, $(/, \mathit{divisor}, 4)$
precise derived-from	artifact	artifact	$(2, \mathit{dividend}, 8)$, $(2, \mathit{divisor}, 4)$

They form two triangles: $(2, 8, /, \mathit{dividend})$ and $(2, 4, /, \mathit{divisor})$. □

In this paper, unless otherwise explicitly stated, we only consider legal OPM graphs. Whenever we refer to a single OPM graph G, we use the names defined in this section to refer to the different constituents of the OPM graph. If we handle more than one OPM graph, for instance graphs G and H, we use superscripts G and H to distinguish their respective constituents. We extend this convention to other concepts related to OPM graphs.

3.2 Temporal models for OPM graphs

The OPM reference specification [20] allows OPM graphs to be decorated with time information for specific time-points, which are meaningful in the context of a computation. Four of these are identified: the beginning of a process, the ending of a process, the instant a process uses an artifact, and the moment a process creates an artifact. Such time information is routinely captured by computer systems. For instance, creation time is readily available from file systems in typical operating systems. HTTP servers and databases logs would usually

[2]The usage role in the use–generate–derive triangle is crucial. We could imagine an extension of Figure 2, in which P uses B in a second role, say s. The triangle of Figure 2 identifies the precise usage of B that affected the output A, here r, whereas, an alternate use of B, with role s, could have not impacted A (for instance, because it took place after A was created).

include the time at which a document or table is read or queried, respectively. Likewise, the beginning and ending time of processes are frequently recorded by job submission systems. The OPM reference specification introduces some constraints between time-points, such as an artifact can only be used after it has been created. In this section, we revisit the notion of time in OPM, by means of a temporal interpretation of a graph, in terms of all the time constraints that it implies (Note 11.6).

A *temporal interpretation* of a legal OPM graph is an assignment of the following time-points to processes, artifacts, and precise used-edges:[3] (Note 11.7)

- for each artifact A, its creation time, denoted by create(A);
- for each process P, its beginning and ending times, denoted by begin(P) and end(P) respectively;
- for each precise used-edge $P \xrightarrow{r} A$, the time when P "read" A in role r, denoted by use(P, r, A).

Formally, we fix some OPM graph G for the remainder of this section. We define the set of *temporal variables* of G, denoted by $\mathit{Vars}(G)$, as follows:

$$\mathit{Vars}(G) = \{\text{create}(A) \mid A \in \mathit{Art}\} \cup \{\text{begin}(P), \text{end}(P) \mid P \in \mathit{Proc}\} \cup \{\text{use}(P, r, A) \mid (P, r, A) \in \mathit{Used}!\}\,.$$

We then define:

Definition 3.4. A *temporal interpretation* of G is a triple $(T, \leq, \tau)$, where

- T is a set, we call its elements *time-points*;
- $\leq$ is a partial order on T;
- τ is a mapping from $\mathit{Vars}(G)$ to T.

When no confusion can arise, we omit T and $\leq$ from the notation and simply denote a temporal interpretation by τ.

Not every temporal interpretation makes sense as a temporal model of G. Indeed, to reflect the dependencies specified in G, the interpretation should satisfy various constraints reflecting these dependencies.

In order to define these constraints formally, we define an *inequality* over G as an expression of the form $u \preceq v$, with $u, v \in \mathit{Vars}(G)$. By a *trivial* inequality we mean an inequality of the form $u \preceq u$. We are now ready to define the set of constraints expressed by a legal OPM graph.

Definition 3.5. The *temporal theory* of G, denoted by Th(G), is the set consisting of all the inequalities stated in the following *axioms*:

AX 1: for each process P, the inequality begin(P) $\preceq$ end(P);

AX 2: for each precise generated-by edge $A \xrightarrow{!} P$ in G, the inequalities begin(P) $\preceq$ create(A) $\preceq$ end(P);

[3] One may wonder why precise generated-by edges do not get a time-point. But, as a matter of fact, they do. For each precise generated-by edge $A \xrightarrow{r} P$, we indeed have a time-point create(A). Since the OPM graph is legal, there can be at most one precise edge emanating from A, so we do not need to specify r and P.

AX 3: for each precise used-edge $P \xrightarrow{r} A$ in G, the three inequalities $\text{begin}(P) \preceq \text{use}(P, r, A)$, $\text{use}(P, r, A) \preceq \text{end}(P)$, and $\text{create}(A) \preceq \text{use}(P, r, A)$;

AX 4: for each imprecise derived-from edge $A \rightarrow B$ in G, the inequality $\text{create}(B) \preceq \text{create}(A)$;

AX 5: for each imprecise generated-by edge $A \rightarrow P$ in G, the inequality $\text{begin}(P) \preceq \text{create}(A)$;

AX 6: for each imprecise used-edge $P \rightarrow A$ in G, the inequality $\text{create}(A) \preceq \text{end}(P)$;

AX 7: for each informed-by edge $P \rightarrow Q$ in G, the inequality $\text{begin}(Q) \preceq \text{end}(P)$;

AX 8: for each $G \triangle (A, B, P, r)$, the inequality $\text{use}(P, r, B) \preceq \text{create}(A)$.

Axioms 1–7 are either obvious (e.g., axiom 1) or are in line with the OPM reference specification (Note 11.8). The eighth axiom, the "triangle axiom", corresponds to the intended usage of OPM by which a precise derived-from edge in a generate–use–derive triangle (A, B, P, r) in G is not redundant, but expresses exactly that P needed to read B in role r before it could generate A (Note 11.9).

Example 3.6. The temporal interpretation of precise edges can be illustrated by a service analogy. Let us consider a translation Web Service P. The service has to be running to receive a translation request A and for the translation result B to be returned; so A is received (used) and B is sent (created) after the beginning of P and before its end; furthermore, B is created after A is received. □

Example 3.7. Referring to Figure 1, the coffee machine operator begins the "Make Coffee" process by cleaning the steam pipe and emptying the coffee filter; once an "empty cup" and "receipt" are available, they are used (precise edge) to generate a "latte" (precise edge). In the same figure, it is unspecified when the "order" is taken, with respect to the beginning of the "Provide Beverages" process; hence, an imprecise used-edge appears in the figure. □

We finally define the temporal models of G as follows. Naturally, a temporal interpretation τ is said to *satisfy* an inequality $u \preceq v$ if $\tau(u) \leq \tau(v)$.

Definition 3.8. A temporal interpretation τ of G is a *temporal model* of G, denoted by $\tau \models \text{Th}(G)$, if it satisfies all inequalities from $\text{Th}(G)$.

Example 3.9. Consider the small OPM graph G shown in Figure 2. Let us use natural numbers with their natural ordering as time-points. Then the two interpretations τ_1 and τ_2, presented in Table 3, are temporal models of G. Temporal model τ_2, which maps all temporal variables to the same time-point, might be generated by a very coarse clock.

Many temporal interpretations of G, however, are not temporal models of G. If, for example, we would modify τ_1 to τ_1' by setting $\tau_1'(\text{end}(P)) = 0$, then Axiom 1 would be violated. Likewise, if we would modify τ_2 to τ_2' by setting $\tau_2'(\text{use}(P, r, B)) = 0$, then Axiom 3 would be violated. Also, if we would modify τ_1 to τ_1'' by setting $\tau_1''(\text{create}(A)) = 0$, then we would violate Axioms 2 and 8. □

Table 3: Two temporal models for the graph shown in Figure 2.

τ_1	variable	value
	create(B)	1
	begin(P)	2
	use(P, r, B)	3
	create(A)	4
	end(P)	5

τ_2	variable	value
	create(B)	1
	begin(P)	1
	use(P, r, B)	1
	create(A)	1
	end(P)	1

Example 3.10. For another example, consider an OPM graph with two artifacts A and B and nothing else (no edges either). Then any possible temporal interpretation qualifies as a model. In particular, in some models τ we have $\tau(\text{create}(A)) < \tau(\text{create}(B))$; in other models we have $\tau(\text{create}(B)) < \tau(\text{create}(A))$; and still in others we have $\tau(\text{create}(A)) = \tau(\text{create}(B))$. This is because the OPM graph does not impose any constraints by the absence of any edges. □

Whilst the temporal semantics is a novel contribution of this paper, the OPM reference specification defines time placeholders in some constructs, and allows them to be filled with time information. These time-decorated constructs correspond to the time variables introduced in this paper. The OPM reference specification does not mandate all time placeholders to be filled. Thus, from a temporal semantics viewpoint, for every decorated construct, the time information found in the placeholder fixes τ for the corresponding variable. If all placeholders are filled, then a single temporal interpretation exists. In general, for every filled placeholder, the number of possible interpretations is reduced.

Now that we have formally defined a temporal model for OPM graphs, we can investigate, in the next section, how we can conduct inference in OPM graphs. Whether there are other, non-temporal, ways to provide a formal semantics for OPM graphs is an interesting direction for further research. (We discuss other efforts in Section 8.)

4 Inference in OPM graphs

The axioms of Definition 3.5 allow us to obtain a number of inequalities over an OPM graph's variables. These inequalities logically imply further inequalities. For a trivial example, in an OPM graph with derived-from edges $A \rightarrow B \rightarrow C$, Axiom 4 gives the inequalities $\text{create}(C) \preceq \text{create}(B)$ and $\text{create}(B) \preceq \text{create}(A)$, which logically imply the further inequality $\text{create}(C) \preceq \text{create}(A)$.

Formally, we define:

Definition 4.1. Let G be a legal OPM graph and let $u, v \in \mathit{Vars}(G)$. The inequality $u \preceq v$ is a *logical consequence* of G, denoted by $\text{Th}(G) \models u \preceq v$, if $u \preceq v$ is satisfied in every temporal model of G.

A general example of logical consequence is provided by the following lemma and proof.

Lemma 4.2. *Let G be a legal OPM graph with artifacts A and B and a precise edge $A \xrightarrow{r} B$ for some role r. Then* $\text{Th}(G) \models \text{create}(B) \preceq \text{create}(A)$.

Before proving this lemma we note that Axiom 4 is almost exactly the same, except that it is stated for an imprecise derived-from edge. Thus, the present lemma shows that the same constraint holds for precise derived-from edges. This constraint did not need to be explicitly given as an axiom because it already logically follows from the given axioms.

Proof. Since G is legal, there exists a process P in G with edges $P \xrightarrow{r} B$ and $A \xrightarrow{!} P$. Let τ be a temporal model of G. By Axiom 3 we have $\tau(\text{create}(B)) \leq \tau(\text{use}(P, r, B))$. By Axiom 2 we have $\tau(\text{use}(P, r, B)) \leq \tau(\text{create}(A))$. We conclude $\tau(\text{create}(B)) \leq \tau(\text{create}(A))$ as desired. □

One may indeed wonder exactly which inequalities logically follow from a given OPM graph. It is well known that an inequality $u \preceq w$ can be inferred from $\text{Th}(G)$ by using repeated applications of the rule of transitivity: "from $u \preceq v$ and $v \preceq w$ we infer $u \preceq w$".[4] However, this way it is hard to relate the newly inferred inequalities to nodes and edges in the graph. Fortunately, we show in Section 4.2 that it is possible to perform temporal inference in a purely graphical manner. We prove in Theorem 4.4 that every possible logical consequence can be directly inferred from the OPM graph by looking for a fixed set of patterns in the graph.

4.1 Edge-inference rules

The cornerstone of our graph-based inference of inequalities is provided by four inference rules that infer new edges in an OPM graph. These four rules are already part of the OPM reference specification [20], except that here we extend them to better take precise edges into account. Inferred edges prove to play an important role in graphical patterns that we introduce to infer inequalities. Moreover, we establish that inference of edges is the only action we need to perform to infer inequalities that do not involve use-variables. (For inequalities involving use-variables, patterns more complicated than a single edge have to be matched in the graph.) We thus provide a justification for the edge inferences introduced in the OPM reference specification.

We first introduce the inference of edges at an intuitive level. Then we define it formally in Definition 4.3. According to the OPM reference specification, the edges present in an OPM graph G denote dependencies. From the given dependencies in G, we can infer derived dependencies. A very intuitive type of inference is to follow chains of derived-from edges. In this section, we define edge-inference rules based on this intuition.

Suppose there is a chain of derived-from edges in G (which can be either precise or imprecise) that starts in an artifact A and ends in an artifact C. We denote this by $A \dashrightarrow C$. Formally, relation $\dashrightarrow$ between two artifacts is nothing else than the transitive closure of *DerivedEdges*. Since A has been indirectly derived from C, we can think of $A \dashrightarrow C$ as an inferred edge, as illustrated in Figure 3(a).

Next we show how to infer generated-by edges. Suppose we have artifacts A and B with $A \dashrightarrow B$ and, in addition, a generated-by edge from B to a process

[4]For a set of inequalities Σ and an inequality φ, φ is a logical consequence of Σ if and only if φ can be inferred from Σ by using transitivity. Ullman [30] presents a self-contained proof for a slightly more general case.

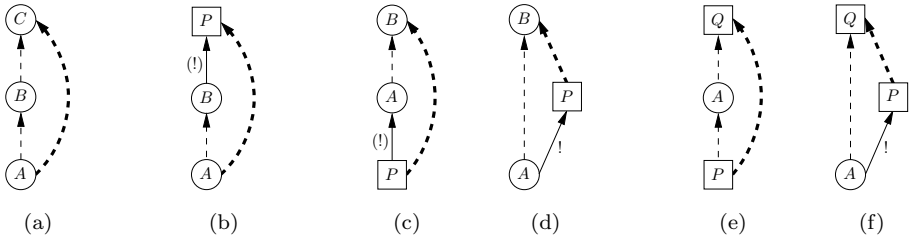

Figure 3: Inference of (a) derived-from, (b) generated-by, (c)–(d) used and (e)–(f) informed-by edges. The bold edges are newly inferred. The edges labeled by "(!)" may be either precise or imprecise.

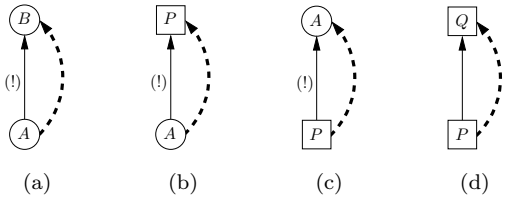

Figure 4: Trivial inference of (a) derived-from, (b) generated-by, (c) used and (d) informed-by edges.

P in G, either a precise edge $B \xrightarrow{!} P$ or an imprecise edge $B \rightarrow P$. Then A has been indirectly generated by P and we can infer an edge $A \dashrightarrow P$, as illustrated in Figure 3(b).

We can infer used-edges as well. Suppose we have artifacts A and B with $A \dashrightarrow B$. In addition, there is a used-edge from a process P to A in G, either precise $P \xrightarrow{!} A$ or imprecise $P \rightarrow A$. Then P has indirectly used B and we can infer an edge $P \dashrightarrow B$, as illustrated in Figure 3(c). Moreover, we can also infer a used-edge in the following situation. Suppose we again have $A \dashrightarrow B$, but now in combination with a precise edge $A \xrightarrow{!} P$ in G. Since A was precisely generated by P, but A has also been indirectly derived from B, we can conclude that P has indirectly used B. Again, we can infer $P \dashrightarrow B$, which we show in Figure 3(d).

Finally, to infer informed-by edges, we can reason as follows. Suppose, for some processes P and Q and an artifact A, we have edges $P \dashrightarrow A$ and $A \dashrightarrow Q$, which are already present in G (either precise or imprecise) or have been previously inferred. Then A represents information that flowed from Q to P and we can infer an edge $P \dashrightarrow Q$, as illustrated in Figure 3(e). Moreover, an informed-by edge can also be inferred in the following case. Suppose we again have $A \dashrightarrow Q$, but now in combination with a precise edge $A \xrightarrow{!} P$ in G. Since A was directly generated by P, but A was also indirectly generated by Q, we can conclude that P was somehow influenced by Q. Again, we can infer $P \dashrightarrow Q$, which we show in Figure 3(f).

There are also trivial inferences for all types of edges, to the effect that an edge that is already present in the graph can always be inferred, as illustrated in Figure 4.

The above discussion is formalized in the following definition. We present

$$\frac{A \to B \text{ in } G \text{ or } A \xrightarrow{!} B \text{ in } G}{G \vdash A \dashrightarrow B} \text{ TRIVIAL DERIVED-FROM}$$

$$\frac{A \to P \text{ in } G \text{ or } A \xrightarrow{!} P \text{ in } G}{G \vdash A \dashrightarrow P} \text{ TRIVIAL GENERATED-BY}$$

$$\frac{P \to A \text{ in } G \text{ or } P \xrightarrow{!} A \text{ in } G}{G \vdash P \dashrightarrow A} \text{ TRIVIAL USED}$$

$$\frac{P \to Q \text{ in } G}{G \vdash P \dashrightarrow Q} \text{ TRIVIAL INFORMED-BY}$$

Figure 5: Trivial edge-inference rules.

$$\frac{G \vdash A \dashrightarrow B \qquad G \vdash B \dashrightarrow C}{G \vdash A \dashrightarrow C} \text{ DERIVED-FROM}$$

$$\frac{G \vdash A \dashrightarrow B \qquad B \to P \text{ in } G \text{ or } B \xrightarrow{!} P \text{ in } G}{G \vdash A \dashrightarrow P} \text{ GENERATED-BY}$$

$$\frac{G \vdash A \dashrightarrow B \qquad P \to A \text{ in } G \text{ or } P \xrightarrow{!} A \text{ in } G \text{ or } A \xrightarrow{!} P \text{ in } G}{G \vdash P \dashrightarrow B} \text{ USED}$$

$$\frac{G \vdash A \dashrightarrow Q \qquad G \vdash P \dashrightarrow A \text{ or } A \xrightarrow{!} P \text{ in } G}{G \vdash P \dashrightarrow Q} \text{ INFORMED-BY}$$

Figure 6: Edge-inference rules.

the rules in a standard notation used in formal logic, where for each rule the premises are stated above a bar, and the conclusion below it.

Definition 4.3 (Edge-inference rules)**.** Let G be a legal OPM graph and let X and Y be two nodes in G. In the following, we define when $X \dashrightarrow Y$ can be *inferred* from G, denoted by $G \vdash X \dashrightarrow Y$. Specifically, let A, B and C be artifacts in G, and let P and Q be processes in G.

We begin by stating four trivial inference rules which mean that if an edge already belongs to G, then that edge can be inferred from G. These rules are presented in Figure 5. Next we define four further inference rules, in cases where at least one of the present edges was previously inferred. These rules are presented in Figure 6.

Note that, as a direct consequence of the above definition, we have the following properties:

- $G \vdash A \dashrightarrow B$ iff (A, B) belongs to the transitive closure of *DerivedEdges*;

- if $G \vdash A \dashrightarrow B$ and $G \vdash B \dashrightarrow P$ then $G \vdash A \dashrightarrow P$;
- if $G \vdash P \dashrightarrow A$ and $G \vdash A \dashrightarrow B$ then $G \vdash P \dashrightarrow B$.

Edge-inference rules introduced in this section allow us to derive new edges from a graph G, noted as $G \vdash X \dashrightarrow Y$, with X and Y two nodes of G. Inferred edges do not belong to the sets of edges identified in Definition 3.1, implying that these edges $X \dashrightarrow Y$ are inferred "outside" G. Thus, the temporal theory of Definition 3.5 does not associate a temporal meaning to these edges, directly. In the next section, we observe that inferred edges have a similar temporal semantics as imprecise edges.

4.2 Characterization of temporal inference

Let us reconsider the axioms of Definition 3.5 that define the temporal semantics of an OPM graph. We see that each axiom is a rule that relates a pattern in the graph to one or more inequalities. For example, Axiom 2 relates the pattern consisting simply of a single edge $A \xrightarrow{!} P$, to the inequalities $\text{begin}(P) \preceq \text{create}(A)$ and $\text{create}(A) \preceq \text{end}(P)$. Axiom 1 even relates the pattern consisting simply of a process node P to the inequality $\text{begin}(P) \preceq \text{end}(P)$. Axiom 8 has a more complicated pattern in the form of a use–generate–derive triangle.

In a similar way, we now introduce ten more such rules. Rules 1–9A–9B are shown in Figure 7. (The figure also includes some axioms, but we explain this after the statement of Theorem 4.4.) An important difference with the axioms, however, is that every dashed edge in a pattern now stands not just for an edge that is present in the graph, but for an edge that can be inferred by the edge-inference rules.

The following theorem states that these rules are *sound* and *complete* in the following sense. The rules are sound in that they represent valid inferences: the inequalities they infer are indeed logical consequences of the axioms in the sense of Definition 4.1. Moreover, the rules are complete in that any inequality that is a logical consequence of the axioms, and that is not already part of the axioms, can be inferred by one of the ten rules.

Theorem 4.4. *Let G be a legal OPM graph and let φ be a nontrivial inequality over the temporal variables of G. Then* $\text{Th}(G) \models \varphi$ *if and only if either (0) φ already belongs to* $\text{Th}(G)$*, or φ matches one of the following inequalities:*

- *Cases* not *involving use-variables:*
  1. $\text{create}(B) \preceq \text{create}(A)$ *with* $G \vdash A \dashrightarrow B$*;*
  2. $\text{begin}(P) \preceq \text{create}(A)$ *with* $G \vdash A \dashrightarrow P$*;*
  3. $\text{create}(A) \preceq \text{end}(P)$ *with* $G \vdash P \dashrightarrow A$*;*
  4. $\text{begin}(Q) \preceq \text{end}(P)$ *with* $G \vdash P \dashrightarrow Q$*;*
- *Cases involving use-variables:*
  5. $\text{create}(B) \preceq \text{use}(P, r, A)$ *with* $P \xrightarrow{r} A$ *in* G *and* $G \vdash A \dashrightarrow B$*;*
  6. $\text{begin}(Q) \preceq \text{use}(P, r, A)$ *with* $P \xrightarrow{r} A$ *in* G *and* $G \vdash A \dashrightarrow Q$*;*
  7. $\text{use}(P, r, C) \preceq \text{create}(A)$ *with* $G \triangle (B, C, P, r)$ *for some* B*, and* $G \vdash A \dashrightarrow B$*;*

8. $\text{use}(P, r, B) \preceq \text{end}(Q)$ *with* $G \triangle (A, B, P, r)$ *for some* A*, and* $G \vdash Q \dashrightarrow A$*;*
9. $\text{use}(P, r, B) \preceq \text{use}(Q, s, A)$ *with* $G \triangle (C, B, P, r)$ *for some* C*, with* $Q \xrightarrow{s} A$ *in* G*, and either (a)* $A = C$ *or (b)* $G \vdash A \dashrightarrow C$*.*

Note that in the above, A, B and C, or P and Q, need not be distinct.

Since Rules 1–4 subsume Axioms 4–7, Figure 7, which includes the remaining axioms, provides a complete picture of the possible logical consequences of an OPM graph in the sense of Definition 4.1. That definition was purely semantic and does not give any concrete algorithm for checking logical consequence. The figure now gives us direct shortcuts from patterns in an OPM graph to its logical consequences.

The inference rules of Figure 7 are an entirely novel characterization of the temporal inferences of OPM since they are sound and complete, in the sense defined in this section. To check that an inequality $u \preceq v$ is logical consequence of a graph, it suffices to select the corresponding pattern in Figure 7, and verify that it is satisfied by the graph (extended with the proper inferred edges). Vice versa, if an inequality $u \preceq v$ is logical consequence of $\text{Th}(G)$, then the corresponding pattern is known to exist in G.

We anticipate that developers can leverage Theorem 4.4 to design reasoners for OPM. So far, reasoners have typically relied on Semantic Web technologies, such as OWL and SRWL, to compute transitive closures of OPM properties [22, 14]. What this theorem shows is that there are logical consequences involving use-variables that cannot be directly represented by edges in OPM graphs.

4.3 Proof of Theorem Theorem 4.4

In this section we present the proof of Theorem 4.4. First, we tackle the soundness property; then, we address the completeness proposition.

4.3.1 Proof of soundness

Let G be a legal OPM graph and let φ be a nontrivial inequality over the temporal variables of G that satisfies the conditions from Theorem 4.4. We have to show that $\text{Th}(G) \models \varphi$. Thereto, let τ be a temporal model of G, i.e., $\tau \models \text{Th}(G)$. We have to show that τ satisfies φ. We inspect the ten possibilities for φ:

(0) if $\varphi \in \text{Th}(G)$, then τ satisfies φ since $\tau \models \text{Th}(G)$.

(1) φ is $\text{create}(B) \preceq \text{create}(A)$ with $G \vdash A \dashrightarrow B$.

As a consequence of Definition 4.3, $G \vdash A \dashrightarrow B$ holds if (A, B) belongs to the transitive closure of *DerivedEdges*. Therefore, there is a path $A_1, A_2, \ldots, A_n$ of derived-from edges from A to B, for some $n \geq 2$ with $A_1 = A$ and $A_n = B$, and with $(A_i, A_{i+1}) \in \textit{DerivedEdges}$, for $i \in \{1, \ldots, n-1\}$. Since every (A_i, A_{i+1}) is an edge in G, we know that $\text{create}(A_{i+1}) \preceq \text{create}(A_i)$ belongs to $\text{Th}(G)$ (Axiom 4 and Lemma 4.2) and is thus satisfied by τ, i.e., $\tau(\text{create}(A_{i+1})) \leq \tau(\text{create}(A_i))$. Hence we also have $\tau(\text{create}(A_n)) \leq \tau(\text{create}(A_1))$, because $\leq$ is a partial order for τ. Thus τ satisfies $\text{create}(B) \preceq \text{create}(A)$.

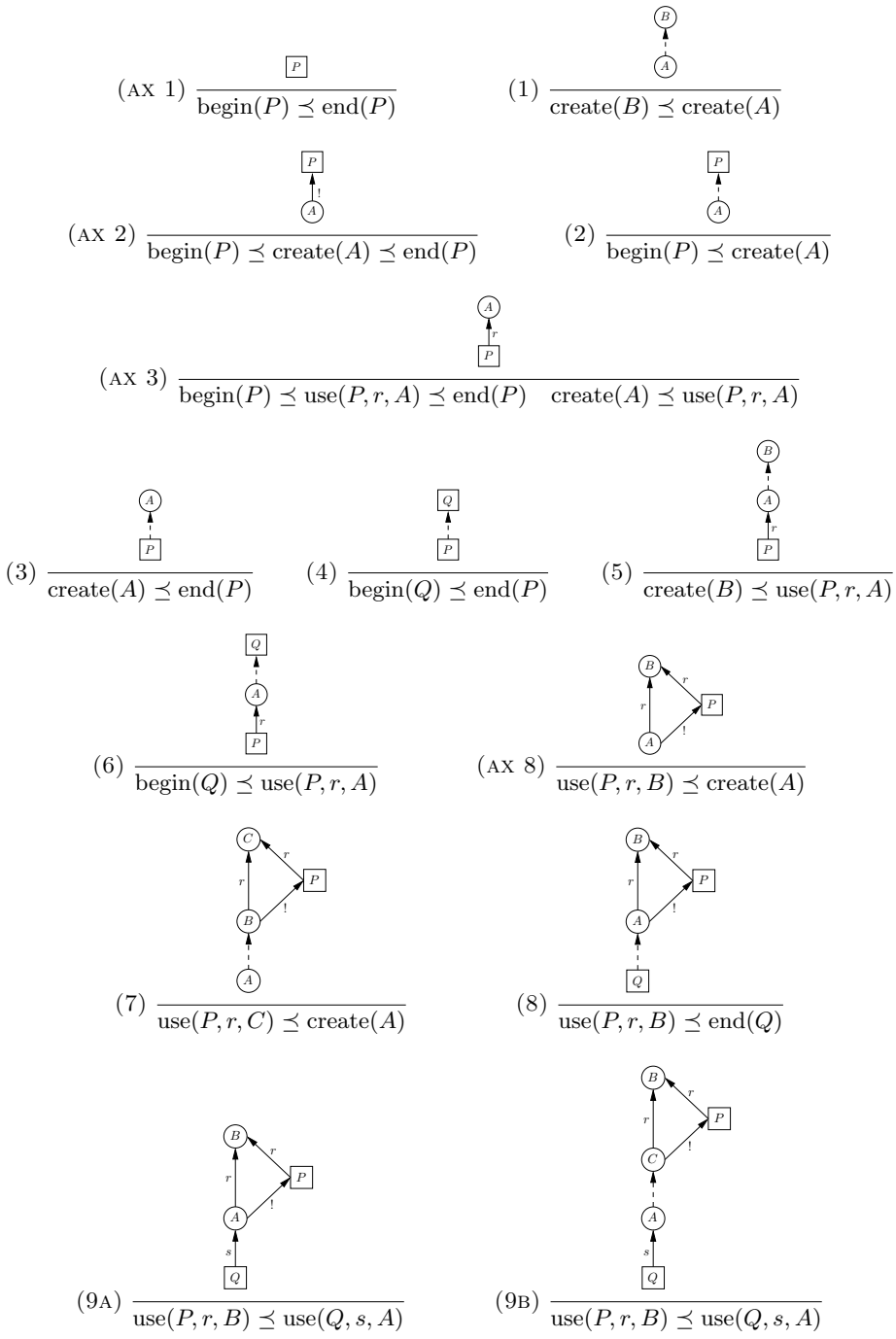

Figure 7: Characterization of temporal inference.

(2) φ is $\text{begin}(P) \preceq \text{create}(A)$ with $G \vdash A \dashrightarrow P$.

By Definition 4.3, $G \vdash A \dashrightarrow P$ if either

a) there is already an edge $A \rightarrow P$ or $A \xrightarrow{!} P$ in G; or

b) there is an artifact B such that $G \vdash A \dashrightarrow B$ and there is an edge $B \rightarrow P$ or $B \xrightarrow{!} P$ in G.

2a) For an edge $A \rightarrow P$ ($A \xrightarrow{!} P$) in G, we know by Axiom 5 (Axiom 2), that $\varphi \in \text{Th}(G)$ and thus τ satisfies φ.

2b) We already know from case 1 that τ satisfies $\text{create}(B) \preceq \text{create}(A)$ for $G \vdash A \dashrightarrow B$, i.e., we have $\tau(\text{create}(B)) \leq \tau(\text{create}(A))$. For an edge $B \rightarrow P$ ($B \xrightarrow{!} P$) in G, we know by Axiom 5 (Axiom 2), that $\text{begin}(P) \preceq \text{create}(B)$ belongs to $\text{Th}(G)$. Therefore τ satisfies $\text{begin}(P) \preceq \text{create}(B)$, i.e., $\tau(\text{begin}(P)) \leq \tau(\text{create}(B))$. Hence $\tau(\text{begin}(P)) \leq \tau(\text{create}(A))$, since $\leq$ is a partial order for τ. We conclude that τ satisfies $\text{begin}(P) \preceq \text{create}(A)$.

(3) φ is $\text{create}(A) \preceq \text{end}(P)$ with $G \vdash P \dashrightarrow A$.

By Definition 4.3, $G \vdash P \dashrightarrow A$ if either

a) there is already an edge $P \rightarrow A$ or $P \xrightarrow{!} A$ in G; or

b) there is an artifact B such that $G \vdash B \dashrightarrow A$ and there is an edge $P \rightarrow B$ or $P \xrightarrow{!} B$ or $B \xrightarrow{!} P$ in G.

3a) For an edge $P \rightarrow A$ ($P \xrightarrow{!} A$) in G, we know by Axiom 6 (Axiom 3) that $\varphi \in \text{Th}(G)$ and thus τ satisfies φ.

3b) We already know from case 1 that τ satisfies $\text{create}(A) \preceq \text{create}(B)$ for $G \vdash B \dashrightarrow A$. For an edge $P \rightarrow B$ ($P \xrightarrow{!} B$) in G, we know by Axiom 6 (Axiom 3) that $\text{create}(B) \preceq \text{end}(P)$ belongs to $\text{Th}(G)$. For an edge $B \xrightarrow{!} P$ in G, we know by Axiom 2 that $\text{create}(B) \preceq \text{end}(P)$ belongs to $\text{Th}(G)$. Therefore, in each case, τ satisfies both $\text{create}(A) \preceq \text{create}(B)$ and $\text{create}(B) \preceq \text{end}(P)$. Hence τ also satisfies $\text{create}(A) \preceq \text{end}(P)$.

(4) φ is $\text{begin}(Q) \preceq \text{end}(P)$ with $G \vdash P \dashrightarrow Q$.

By Definition 4.3, $G \vdash P \dashrightarrow Q$ if either

a) there is already an edge $P \rightarrow Q$ in G; or

b) there is an artifact A such that $G \vdash A \dashrightarrow Q$, and either $G \vdash P \dashrightarrow A$ or there is an edge $A \xrightarrow{!} P$ in G.

4a) For an edge $P \rightarrow Q$ in G, we know by Axiom 7 that $\varphi \in \text{Th}(G)$ and thus τ satisfies φ.

4b) We already know from case 2 that τ satisfies $\text{begin}(Q) \preceq \text{create}(A)$ for $G \vdash A \dashrightarrow Q$. We also know from case (3) that τ satisfies $\text{create}(A) \preceq \text{end}(P)$ for $G \vdash P \dashrightarrow A$. For an edge $A \xrightarrow{!} P$ in G, we know by Axiom 2 that $\text{create}(A) \preceq \text{end}(P)$ belongs to $\text{Th}(G)$. Therefore, in each case, τ satisfies both $\text{begin}(Q) \preceq \text{create}(A)$ and $\text{create}(A) \preceq \text{end}(P)$. Thus τ also satisfies $\text{begin}(Q) \preceq \text{end}(P)$.

(5) φ is $\text{create}(B) \preceq \text{use}(P, r, A)$ with $P \xrightarrow{r} A$ in G and $G \vdash A \dashrightarrow B$.

We already know from case 1 that τ satisfies $\text{create}(B) \preceq \text{create}(A)$ for $G \vdash A \dashrightarrow B$. For edge $P \xrightarrow{r} A$ in G we know, by Axiom 3, that $\text{create}(A) \preceq \text{use}(P, r, A)$ belongs to $\text{Th}(G)$, and is thus satisfied by τ. Therefore, τ also satisfies $\text{create}(B) \preceq \text{use}(P, r, A)$.

(6) φ is $\text{begin}(Q) \preceq \text{use}(P, r, A)$ with $P \xrightarrow{r} A$ in G and $G \vdash A \dashrightarrow Q$.

We already know from case 2 that τ satisfies $\text{begin}(Q) \preceq \text{create}(A)$ for $G \vdash A \dashrightarrow Q$. For edge $P \xrightarrow{r} A$ in G, we know, by Axiom 3, that $\text{create}(A) \preceq \text{use}(P, r, A)$ belongs to $\text{Th}(G)$, and is thus satisfied by τ. Thus, τ also satisfies $\text{begin}(Q) \preceq \text{use}(P, r, A)$.

(7) φ is $\text{use}(P, r, C) \preceq \text{create}(A)$ with $G \triangle (B, C, P, r)$ and $G \vdash A \dashrightarrow B$.

We already know from case 1 that τ satisfies $\text{create}(B) \preceq \text{create}(A)$ for $G \vdash A \dashrightarrow B$. From $G \triangle (B, C, P, r)$ we know, by Axiom 8, that $\text{use}(P, r, C) \preceq \text{create}(B)$ belongs to $\text{Th}(G)$, and is thus satisfied by τ. Therefore, τ also satisfies $\text{use}(P, r, C) \preceq \text{create}(A)$.

(8) φ is $\text{use}(P, r, B) \preceq \text{end}(Q)$ with $G \triangle (A, B, P, r)$ and $G \vdash Q \dashrightarrow A$.

We already know from case 3 that τ satisfies $\text{create}(A) \preceq \text{end}(Q)$ for $G \vdash Q \dashrightarrow A$. From $G \triangle (A, B, P, r)$ we know, by Axiom 8, that $\text{use}(P, r, B) \preceq \text{create}(A)$ belongs to $\text{Th}(G)$, and is thus satisfied by τ. Hence, τ also satisfies $\text{use}(P, r, B) \preceq \text{end}(Q)$.

(9) φ is $\text{use}(P, r, B) \preceq \text{use}(Q, s, A)$ with $G \triangle (C, B, P, r)$ in G, $Q \xrightarrow{s} A$ in G, and either (a) $A = C$ or (b) $G \vdash A \dashrightarrow C$.

We already know from case 1 that τ satisfies $\text{create}(C) \preceq \text{create}(A)$ for $G \vdash A \dashrightarrow C$ (9b). If $A = C$ (9a) then, obviously, $\tau(A) = \tau(C)$, and τ still satisfies $\text{create}(C) \preceq \text{create}(A)$. For edge $Q \xrightarrow{s} A$ in G, we know, by Axiom 3, that $\text{create}(A) \preceq \text{use}(Q, s, A)$ belongs to $\text{Th}(G)$, and is thus satisfied by τ. From $G \triangle (C, B, P, r)$ we know, by Axiom 8S, that $\text{use}(P, r, B) \preceq \text{create}(C)$ belongs to $\text{Th}(G)$, hence is satisfied by τ. Therefore, we have $\text{use}(P, r, B) \preceq \text{create}(C) \preceq \text{create}(A) \preceq \text{use}(Q, s, A)$. We conclude that τ also satisfies $\text{use}(P, r, B) \preceq \text{use}(Q, s, A)$.

4.3.2 Proof of completeness

Let G be a legal OPM graph and let φ be a nontrivial inequality over the temporal variables of G such that $\text{Th}(G) \models \varphi$. We must show that $\varphi \in \text{Th}(G)$ or φ matches one of the cases 1–9 of Theorem 4.4.

It is well known [30] that φ can be inferred from $\text{Th}(G)$ by using repeated applications of the rule of transitivity: "from $u \preceq v$ and $v \preceq w$ infer $u \preceq w$." We proceed by induction on the number of applications of the transitivity rule.

If φ can be inferred by zero applications, then φ is already in $\text{Th}(G)$ and we are done, as this corresponds to case 0 of the theorem.

Now consider an application of transitivity inferring φ of the form $u \preceq w$ from $u \preceq v \preceq w$, where, by induction, the theorem can already be assumed to hold for the inequalities $u \preceq v$ and $v \preceq w$. Since begin-variables (end-variables) never appear on the right-hand (left-hand) side of an inequality, v cannot be a

begin-variable (end-variable). That leaves us with two cases, with v being either a create- or a use-variable.

Case v is a create-variable Let v be a create-variable, say create(A_v). Let us list the possibilities for u and note the relevant properties:

(a) u is also a create-variable, say create(A_u). By induction, we know that the inequality $u \preceq v$ either already belongs to Th(G), so there is an edge $A_v \to A_u$ in G (Axiom 4), or the inequality corresponds to case 1 of the theorem, therefore $G \vdash A_v \dashrightarrow A_u$. In either case we have $G \vdash A_v \dashrightarrow A_u$.

(b) u is a begin-variable, say begin(P_u). By induction, $u \preceq v$ either belongs to Th(G), so there is an edge $A_v \xrightarrow{!} P_u$ in G (Axiom 2) or $A_v \to P_u$ in G (Axiom 5); or $u \preceq v$ corresponds to case 2 of the theorem, therefore $G \vdash A_v \dashrightarrow P_u$. In either case we have $G \vdash A_v \dashrightarrow P_u$.

(c) u is a use-variable, say use(P_u, r_u, A_u). By induction, $u \preceq v$ either (c1) belongs to Th(G), so there is some use–generate–derive triangle (A_v, A_u, P_u, r_u) in G (Axiom 8); or (c2) $u \preceq v$ corresponds to case 7 of the theorem, thus there is a use–generate–derive triangle (A'_v, A_u, P_u, r_u) in G with $G \vdash A_v \dashrightarrow A'_v$.

We also list the possibilities for w and their relevant properties:

(d) w is also a create-variable, say create(A_w). By the induction hypothesis applied to $v \preceq w$, reasoning similarly as in case (a) above, we have $G \vdash A_w \dashrightarrow A_v$.

(e) w is an end-variable, say end(P_w). By induction, $v \preceq w$ either belongs to Th(G), so there is an edge $A_v \xrightarrow{!} P_w$ in G (Axiom 2) or $P_w \to A_v$ in G (Axiom 6); or $v \preceq w$ corresponds to case 3 of the theorem, therefore $G \vdash P_w \dashrightarrow A_v$. We have thus either (e1) $A_v \xrightarrow{!} P_w$ in G or (e2) $G \vdash P_w \dashrightarrow A_v$.

(f) w is a use-variable, say use(P_w, r_w, A_w). This necessitates the presence of edge $P_w \xrightarrow{r_w} A_w$ in G. By induction, the inequality $v \preceq w$ either (f1) belongs to Th(G), so that $A_w = A_v$ (Axiom 3); or (f2) $v \preceq w$ corresponds to case 5 of the theorem, thus $G \vdash A_w \dashrightarrow A_v$.

We can now inspect the nine possible combinations:

(ad) φ is create(A_u) $\preceq$ create(A_w). From $G \vdash A_v \dashrightarrow A_u$ and $G \vdash A_w \dashrightarrow A_v$ we infer $G \vdash A_w \dashrightarrow A_u$, which matches case 1 of the theorem.

(ae) φ is create(A_u) $\preceq$ end(P_w). From $G \vdash A_v \dashrightarrow A_u$ and either $A_v \xrightarrow{!} P_w$ in G or $G \vdash P_w \dashrightarrow A_v$ we infer $G \vdash P_w \dashrightarrow A_u$, which matches case 3 of the theorem.

(af) φ is create(A_u) $\preceq$ use(P_w, r_w, A_w) with $P_w \xrightarrow{r_w} A_w$ in G. In case f1, we have $G \vdash A_v \dashrightarrow A_u$ and $A_v = A_w$, so the case corresponds to case 5 of the theorem. In case f2, we infer $G \vdash A_w \dashrightarrow A_u$ from $G \vdash A_w \dashrightarrow A_v$ and $G \vdash A_v \dashrightarrow A_u$, which again matches case 5 of the theorem.

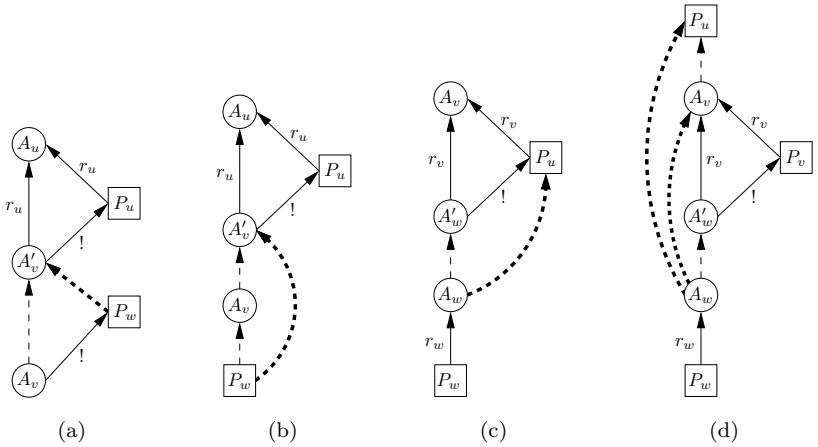

Figure 8: Proof of the completeness of Theorem 4.4 for cases (a) c2 with e1, (b) c2 with e2, (c) h1 with l, and (d) h2 with l. The bold edges are newly inferred.

(bd) φ is $\text{begin}(P_u) \preceq \text{create}(A_w)$. From $G \vdash A_v \dashrightarrow P_u$ and $G \vdash A_w \dashrightarrow A_v$ we infer $G \vdash A_w \dashrightarrow P_u$, which corresponds to case 2 of the theorem.

(be) φ is $\text{begin}(P_u) \preceq \text{end}(P_w)$. From $G \vdash A_v \dashrightarrow P_u$ and either $A_v \xrightarrow{!} P_w$ in G or $G \vdash P_w \dashrightarrow A_v$ we infer $G \vdash P_w \dashrightarrow P_u$, which matches case 4 of the theorem.

(bf) φ is $\text{begin}(P_u) \preceq \text{use}(P_w, r_w, A_w)$ with $P_w \xrightarrow{r_w} A_w$ in G. In case f1, we have $G \vdash A_v \dashrightarrow P_u$ and $A_v = A_w$, so the case corresponds to case 6 of the theorem. In case f2, we infer $G \vdash A_w \dashrightarrow P_u$ from $G \vdash A_w \dashrightarrow A_v$ and $G \vdash A_v \dashrightarrow P_u$, which again matches case 6 of the theorem.

(cd) φ is $\text{use}(P_u, r_u, A_u) \preceq \text{create}(A_w)$. Case c1 corresponds directly to case 7 of the theorem. In case c2, we infer $G \vdash A_w \dashrightarrow A'_v$ from $G \vdash A_v \dashrightarrow A'_v$ and $G \vdash A_w \dashrightarrow A_v$, which again matches case 7 of the theorem.

(ce) φ is $\text{use}(P_u, r_u, A_u) \preceq \text{end}(P_w)$. First, consider case c1 together with e1. Since $G \triangle (A_v, A_u, P_u, r_u)$, P_u and P_w must be equal because G is legal. In this case the inequality holds by Axiom 3. Case c1 together with e2 corresponds directly to case 8 of the theorem. Finally, in case c2, from $G \vdash A_v \dashrightarrow A'_v$, and either $A_v \xrightarrow{!} P_w$ (from e1, see Figure 8(a)) or $G \vdash P_w \dashrightarrow A_v$ (from e2, see Figure 8(b)) we infer $G \vdash P_w \dashrightarrow A'_v$, which matches case 8 of the theorem.

(cf) φ is $\text{use}(P_u, r_u, A_u) \preceq \text{use}(P_w, r_w, A_w)$. Case c1 together with f1 corresponds directly to case 9a of the theorem. Case c1 together with f2 matches case 9b of the theorem. The same holds for c2 together with f1. In case c2 together with f2 we infer $G \vdash A_w \dashrightarrow A'_v$ from $G \vdash A_w \dashrightarrow A_v$ and $G \vdash A_v \dashrightarrow A'_v$, which again matches case 9b of the theorem.

Case v is a use-variable

Let v be a use-variable, say $\mathrm{use}(P_v, r_v, A_v)$. Note that this necessitates the presence of the edge $P_v \xrightarrow{r_v} A_v$ in G. Let us list the possibilities for u and note the relevant properties:

(g) u is a create-variable, say $\mathrm{create}(A_u)$. By induction, we know that the inequality $u \preceq v$ either (g1) already belongs to $\mathrm{Th}(G)$, so A_u equals A_v with the edge $P_v \xrightarrow{r_v} A_v$ in G (Axiom 3); or (g2) the inequality corresponds to case 5 of the theorem, therefore $G \vdash A_v \dashrightarrow A_u$ with the edge $P_v \xrightarrow{r_v} A_v$ in G.

(h) u is a begin-variable, say $\mathrm{begin}(P_u)$. By induction, $u \preceq v$ either (h1) already belongs to $\mathrm{Th}(G)$, thus P_u equals P_v with the edge $P_v \xrightarrow{r_v} A_v$ in G (Axiom 3); or (h2) the inequality corresponds to case 6 of the theorem, therefore $G \vdash A_v \dashrightarrow P_u$ with the edge $P_v \xrightarrow{r_v} A_v$ in G.

(i) u is also a use-variable, say $\mathrm{use}(P_u, r_u, A_u)$. By induction, we know that the inequality $u \preceq v$ can only correspond to case 9 of the theorem, therefore we have some triangle (A'_v, A_u, P_u, r_u) in G with the edge $P_v \xrightarrow{r_v} A_v$ in G, and either $A_v = A'_v$ or $G \vdash A_v \dashrightarrow A'_v$.

We also list the possibilities for w and their relevant properties:

(j) w is a create-variable, say $\mathrm{create}(A_w)$. By induction, $v \preceq w$ either (j1) already belongs to $\mathrm{Th}(G)$, so there is some use–generate–derive triangle (A_w, A_v, P_v, r_v) in G (Axiom 8); or (j2) the inequality corresponds to case 7 of the theorem, and there is a use–generate–derive triangle (A'_w, A_v, P_v, r_v) in G with $G \vdash A_w \dashrightarrow A'_w$. Note that in both cases we can infer $G \vdash A_w \dashrightarrow A_v$. Indeed, in case j1 we have the edge $A_w \xrightarrow{r_v} A_v$ in G. In case j2 we have $G \vdash A_w \dashrightarrow A'_w$ and the edge $A'_w \xrightarrow{r_v} A_v$ in G.

(k) w is an end-variable, say $\mathrm{end}(P_w)$. By induction, $v \preceq w$ either (k1) already belongs to $\mathrm{Th}(G)$, so P_w equals P_v with the edge $P_v \xrightarrow{r_v} A_v$ in G (Axiom 3); or (k2) corresponds to case 8 of the theorem, thus there is some use–generate–derive triangle (A_w, A_v, P_v, r_v) in G with $G \vdash P_w \dashrightarrow A_w$. Note that in both cases we can infer $G \vdash P_w \dashrightarrow A_v$. Indeed, in case k1 we have the edge $P_w \xrightarrow{r_v} A_v$ in G. In case k2 we have $G \vdash P_w \dashrightarrow A_w$ and the edge $A_w \xrightarrow{r_v} A_v$ in G.

(l) w is also a use-variable, say $\mathrm{use}(P_w, r_w, A_w)$. By induction, we know that the inequality $v \preceq w$ can only correspond to case 9 of the theorem, so there is some use–generate–derive triangle (A'_w, A_v, P_v, r_v) in G with $P_w \xrightarrow{r_w} A_w$ in G, and either $A_w = A'_w$ or $G \vdash A_w \dashrightarrow A'_w$. Note that in both cases we can infer $G \vdash A_w \dashrightarrow A_v$ by the edge $A'_w \xrightarrow{r_v} A_v$ in the triangle.

We can now inspect the nine possible combinations:

(gj) φ is $\mathrm{create}(A_u) \preceq \mathrm{create}(A_w)$. In case g1, we have $G \vdash A_w \dashrightarrow A_v$ and $A_v = A_u$, so the case corresponds directly to case 1 of the theorem. In case g2, we have $G \vdash A_w \dashrightarrow A_v$ and $G \vdash A_v \dashrightarrow A_u$, so we can infer $G \vdash A_w \dashrightarrow A_u$, which, again, matches case 1 of the theorem.

(gk) φ is create$(A_u) \preceq$ end(P_w). In case g1 together with k1, we have $A_v = A_u$, $P_v = P_w$, and the edge $P_v \xrightarrow{r_v} A_v$ in G, so the case corresponds directly to case 3 of the theorem. In case g2 together with k1, we have $P_v = P_w$ with the edge $P_v \xrightarrow{r_v} A_v$ in G. From the latter and $G \vdash A_v \dashrightarrow A_u$, we infer $G \vdash P_w \dashrightarrow A_u$, which again matches case 3 of the theorem. In case g1 together with k2, we have $G \vdash P_w \dashrightarrow A_v$ and $A_v = A_u$, so the case corresponds directly to case 3 of the theorem. In case g2 together with k2, we infer $G \vdash P_w \dashrightarrow A_u$ from $G \vdash P_w \dashrightarrow A_v$ and $G \vdash A_v \dashrightarrow A_u$. Hence the case again matches case 3 of the theorem.

(gl) φ is create$(A_u) \preceq$ use(P_w, r_w, A_w). In case g1, we have $G \vdash A_w \dashrightarrow A_v$ and $A_v = A_u$, so the case corresponds directly to case 5 of the theorem. In case g2, we infer $G \vdash A_w \dashrightarrow A_u$ from $G \vdash A_w \dashrightarrow A_v$ and $G \vdash A_v \dashrightarrow A_u$, which matches case 5 of the theorem.

(hj) φ is begin$(P_u) \preceq$ create(A_w). By case j, we infer $G \vdash A_w \dashrightarrow P_v$. (Indeed, in case j1 we easily infer $G \vdash A_w \dashrightarrow P_v$. In case j2 we also infer $G \vdash A_w \dashrightarrow P_v$ from $G \vdash A_w \dashrightarrow A'_w$ and $A'_w \xrightarrow{!} P_v$ in G.) Now in case h1 we have $P_v = P_u$, so the case corresponds directly to case 2 of the theorem. In case h2 we have $G \vdash A_w \dashrightarrow A_v$ and $G \vdash A_v \dashrightarrow P_u$, so we can infer $G \vdash A_w \dashrightarrow P_u$. Thus the case again matches case 2 of the theorem.

(hk) φ is begin$(P_u) \preceq$ end(P_w). In case h1 together with k1, we have $P_u = P_v = P_w$, so the inequality trivially holds (Axiom 1). In case h1 together with k2, we have $P_v = P_u$, and we infer $G \vdash P_w \dashrightarrow P_v$ from $G \vdash P_w \dashrightarrow A_w$ and $G \vdash A_w \dashrightarrow P_v$. Thus the case corresponds to case 4 of the theorem. In case h2 we have $G \vdash P_w \dashrightarrow A_v$, which combined with $G \vdash A_v \dashrightarrow P_u$, yields $G \vdash P_w \dashrightarrow P_u$. Hence the case again matches case 4 of the theorem.

(hl) φ is begin$(P_u) \preceq$ use(P_w, r_w, A_w). By case l, we infer $G \vdash A_w \dashrightarrow P_v$ from $A'_w \xrightarrow{!} P_v$ in G, and either $A_w = A'_w$ or $G \vdash A_w \dashrightarrow A'_w$. Also, there is an edge $P_w \xrightarrow{r_w} A_w$ in G, and $G \vdash A_w \dashrightarrow A_v$. In case h1 (see Figure 8(c)) we have $P_v = P_u$, so the case corresponds to case 6 of the theorem. In case h2 (see Figure 8(d)), from $G \vdash A_w \dashrightarrow A_v$ and $G \vdash A_v \dashrightarrow P_u$, we infer $G \vdash A_w \dashrightarrow P_u$. Hence the case again matches case 6 of the theorem.

(ij) φ is use$(P_u, r_u, A_u) \preceq$ create(A_w). We infer $G \vdash A_w \dashrightarrow A'_v$ from $G \vdash A_w \dashrightarrow A_v$, and either $A_v = A'_v$ or $G \vdash A_v \dashrightarrow A'_v$. Together with $G \triangle (A'_v, A_u, P_u, r_u)$, the case corresponds to case 7 of the theorem.

(ik) φ is use$(P_u, r_u, A_u) \preceq$ end(P_w). We already have $G \triangle (A'_v, A_u, P_u, r_u)$. In case k1, we have $P_w = P_v$. We infer $G \vdash P_v \dashrightarrow A'_v$ from the edge $P_v \xrightarrow{r_v} A_v$ in G, and either $A_v = A'_v$ or $G \vdash A_v \dashrightarrow A'_v$. The case thus matches case 8 of the theorem. In case k2, we infer $G \vdash P_w \dashrightarrow A'_v$ from $G \vdash P_w \dashrightarrow A_w$, $A_w \xrightarrow{r_v} A_v$ in G, and either $A_v = A'_v$ or $G \vdash A_v \dashrightarrow A'_v$. Hence the case again matches case 8 of the theorem.

(il) φ is use$(P_u, r_u, A_u) \preceq$ use(P_w, r_w, A_w). We already have $P_w \xrightarrow{r_w} A_w$ in G and $G \triangle (A'_v, A_u, P_u, r_u)$. We additionally infer $G \vdash A_w \dashrightarrow A'_v$

from $G \vdash A_w \dashrightarrow A_v$, and either $A_v = A'_v$ or $G \vdash A_v \dashrightarrow A'_v$. Hence the case matches case 9b of the theorem.

4.4 About no-use inequalities

The Open Provenance Model reference specification defines edges adjacent to artifacts in terms of the creation of the artifact, with respect to the creation of another artifact, or the beginning and ending of a process. There is some value in considering a temporal theory that ignores use time-points, since the theory becomes simpler (though it is unable to tell us anything about usage of artifacts). In this case, it is worth characterizing temporal inference in the context of this simpler theory.

First we state a remarkable corollary, after introducing the following definition.

Definition 4.5. If in an inequality φ of the form $u \preceq v$, neither u nor v is a use-variable, then we call φ a *no-use* inequality.

As a corollary to Theorem 4.4, we obtain the following completeness result for edge inference, as far as no-use inequalities are concerned. Note that the edge-inference rules are present as Rules 1–4 in Figure 7.

Corollary 4.6. *Let G be a legal OPM graph and let φ be a no-use inequality. Then* $\mathrm{Th}(G) \models \varphi$ *if and only if φ can be inferred using Axioms 1–2 and Rules 1–4 in Figure 7.*

Proof. It is clear from Theorem 4.4 that if φ can be inferred using Axioms 1–2 and Rules 1–4, then $\mathrm{Th}(G) \models \varphi$. For the other direction, assume that $\mathrm{Th}(G) \models \varphi$ holds. Then we know by Theorem 4.4 that φ can be inferred by the axioms and rules presented in Figure 7. By examination of these axioms and rules, however, we notice that Axioms 1–2 and Rules 1–4 are the only ones that infer no-use inequalities. □

It is interesting to note, that when dealing with no-use inequalities, we do not need the full temporal theory of an OPM graph. We start with a small generalization of Definitions 3.8 and 4.1.

Definition 4.7. Let G be a legal OPM graph and let $u, v \in \mathit{Vars}(G)$. Let Σ be a subset of $\mathrm{Th}(G)$. Any temporal interpretation that satisfies all inequalities of Σ is called a *temporal model* of Σ. Furthermore, the inequality $u \preceq v$ is a *logical consequence* of Σ, denoted by $\Sigma \models u \preceq v$, if $u \preceq v$ is satisfied in every temporal model of Σ.

We can now select, for a given OPM graph G, only the no-use inequalities from its temporal theory.

Definition 4.8. For a legal OPM graph G, we define the *no-use temporal theory* of G, denoted by $\mathrm{Th}^{\text{no-use}}(G)$, as follows:

$$\begin{aligned}\mathrm{Th}^{\text{no-use}}(G) = \{&\varphi \in \mathrm{Th}(G) \mid \varphi \text{ is a no-use inequality}\} \\ &\cup \left\{\mathrm{create}(A) \preceq \mathrm{end}(P) \mid P \xrightarrow{!} A \text{ in } G\right\} \\ &\cup \left\{\mathrm{create}(B) \preceq \mathrm{create}(A) \mid A \xrightarrow{!} B \text{ in } G\right\}.\end{aligned}$$

The intuition is that $\mathrm{Th}^{\text{no-use}}(G)$ does not contain Axioms 3 and 8, and enforces Axioms 4 and 6 for precise and imprecise edges alike.[5]

We can now observe that use-variables do not influence the no-use inequalities that are logical consequences of $\mathrm{Th}(G)$.

Proposition 4.9. *Let G be a legal OPM graph and let φ be a no-use inequality. Then $\mathrm{Th}(G) \models \varphi$ if and only if $\mathrm{Th}^{no\text{-}use}(G) \models \varphi$.*

Proof. Since any temporal model τ of $\mathrm{Th}(G)$ is also a temporal model of $\mathrm{Th}^{\text{no-use}}(G)$, the if-direction is immediate. For the only-if direction, let τ be a temporal model of $\mathrm{Th}^{\text{no-use}}(G)$. We try to extend τ to τ' in such a way that τ' is a temporal model of $\mathrm{Th}(G)$. For every no-use variable u simply put $\tau'(u) = \tau(u)$. Now we have to find suitable values for:

- $\mathrm{use}(P, r, A)$ for each $P \xrightarrow{r} A$ in G, so that Axiom 3 is satisfied, and
- $\mathrm{use}(P, r, B)$ for each $G \triangle (A, B, P, r)$, so that Axiom 8 is satisfied.

It is easy to verify that Axiom 3 holds if $\tau'(\mathrm{use}(P, r, A))$ equals the maximum of $\tau(\mathrm{create}(A))$ and $\tau(\mathrm{begin}(P))$. Likewise, Axiom 8 holds if $\tau'(\mathrm{use}(P, r, B))$ equals the maximum of $\tau(\mathrm{create}(B))$ and $\tau(\mathrm{begin}(P))$. Thus τ' satisfies all eight axioms and is a temporal model of $\mathrm{Th}(G)$. We know thus that τ' satisfies φ. Since φ is a no-use inequality, and τ' and τ coincide on all variables used in no-use inequalities, τ also satisfies φ. □

The above proposition together with Corollary 4.6 yields the following:

Corollary 4.10. *Let G be a legal OPM graph and let φ be a no-use inequality. Then $\mathrm{Th}^{no\text{-}use}(G) \models \varphi$ if and only if φ can be inferred using Axioms 1–2 and Rules 1–4 in Figure 7.*

This section provides a remarkable result since it establishes the completeness of edge inferences (Rules 1–4 in Figure 7) for no-use inequalities. Furthermore, reasoning with use time-points does not allow us to derive any new inequality about no-use variables. We envisage this result to be leveraged by developers of reasoners for OPM, since it offers opportunities to optimize reasoners, by reducing the number of time-points to reason over, focusing on no-use variables in a first phase, and dealing efficiently with use-variables afterwards.

5 Operations on OPM graphs

The reason for capturing provenance is that it can be used to address a variety of use cases [16]. To this end, one needs to collect provenance information from potentially different sources, combine and process it in multiple ways. It is therefore useful to define operations on OPM graphs, which we anticipate can become part of "provenance toolkits".

[5] Note that in the full theory $\mathrm{Th}(G)$, the no-use inequality $\mathrm{create}(A) \preceq \mathrm{end}(P)$ for $P \xrightarrow{!} A$ in G is implied by Axiom 3, but since we omit this axiom, we need to recover the inequality in axiom 6. Likewise, the no-use inequality $\mathrm{create}(B) \preceq \mathrm{create}(A)$ for $A \xrightarrow{!} B$ in G is provided by Lemma 4.2. Since the proof of the lemma utilizes use-variables, the lemma doesn't hold anymore and we need to recover the inequality in axiom 4.

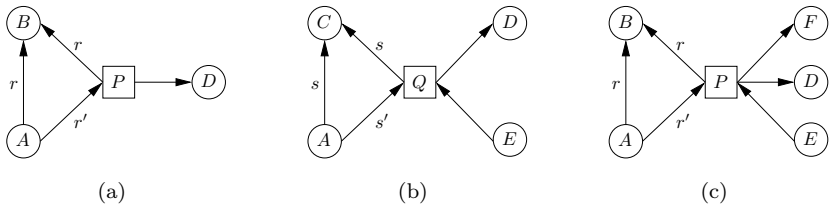

Figure 9: Three legal OPM graphs.

When two OPM graphs are obtained from different sources, a reasoner may want to take their union, if it ascertains they relate to some common entities. Given two OPM graphs, an intersection operation helps identify their common elements. Different sources may use different identifiers for graph nodes; thus, to be able to compute meaningful union and intersection, it may be required to rename some nodes, before performing these operations. In this section, we formalize notions of subgraph, union, intersection, and renaming and merging.

Let us fix two OPM graphs G and H for use in this section. Neither G nor H have to be legal.

Definition 5.1 (Subgraph)**.** H is a *subgraph* of G if every constituent of H is a subset of the corresponding constituent of G. Formally:

- $Art^H \subseteq Art^G$,
- $Proc^H \subseteq Proc^G$,
- $Roles^H \subseteq Roles^G$,
- $GeneratedBy!^H \subseteq GeneratedBy!^G$,
- $Used!^H \subseteq Used!^G$,
- $DerivedFrom!^H \subseteq DerivedFrom!^G$,
- $GeneratedBy^H \subseteq GeneratedBy^G$,
- $Used^H \subseteq Used^G$,
- $DerivedFrom^H \subseteq DerivedFrom^G$,
- $InformedBy^H \subseteq InformedBy^G$.

Note that a subgraph of a legal OPM graph may not be legal. For example, the graph presented in Figure 9(a) is legal, whereas its subgraph composed of nodes A, B, P, role r, and edges $A \xrightarrow{r} B$ and $A \xrightarrow{r'} P$ is not legal, since the use–generate–derive triangle (A, B, P, r) is not complete.

Definition 5.2 (Union)**.** The *union* of G and H, denoted by $G \cup H$, is the OPM graph where each constituent equals the union of the corresponding constituents in G and H. Formally:

- $Art^{G \cup H} = Art^G \cup Art^H$,
- $Proc^{G \cup H} = Proc^G \cup Proc^H$,

- $Roles^{G \cup H} = Roles^G \cup Roles^H$,
- $GeneratedBy!^{G \cup H} = GeneratedBy!^G \cup GeneratedBy!^H$,
- $Used!^{G \cup H} = Used!^G \cup Used!^H$,
- $DerivedFrom!^{G \cup H} = DerivedFrom!^G \cup DerivedFrom!^H$,
- $GeneratedBy^{G \cup H} = GeneratedBy^G \cup GeneratedBy^H$,
- $Used^{G \cup H} = Used^G \cup Used^H$,
- $DerivedFrom^{G \cup H} = DerivedFrom^G \cup DerivedFrom^H$,
- $InformedBy^{G \cup H} = InformedBy^G \cup InformedBy^H$.

Note that the union of two legal OPM graphs may not be legal. For instance, the graph presented in Figure 9(a) is legal, and so is the graph in Figure 9(b). The union of these two graphs, however, is not legal, since in the union A has two different precise generated-by edges: $A \xrightarrow{r'} P$ and $A \xrightarrow{s'} Q$.

Definition 5.3 (Intersection)**.** The *intersection* of G and H, denoted by $G \cap H$, is the OPM graph where each constituent equals the intersection of the corresponding constituents in G and H. Formally:

- $Art^{G \cap H} = Art^G \cap Art^H$,
- $Proc^{G \cap H} = Proc^G \cap Proc^H$,
- $Roles^{G \cap H} = Roles^G \cap Roles^H$,
- $GeneratedBy!^{G \cap H} = GeneratedBy!^G \cap GeneratedBy!^H$,
- $Used!^{G \cap H} = Used!^G \cap Used!^H$,
- $DerivedFrom!^{G \cap H} = DerivedFrom!^G \cap DerivedFrom!^H$,
- $GeneratedBy^{G \cap H} = GeneratedBy^G \cap GeneratedBy^H$,
- $Used^{G \cap H} = Used^G \cap Used^H$,
- $DerivedFrom^{G \cap H} = DerivedFrom^G \cap DerivedFrom^H$,
- $InformedBy^{G \cap H} = InformedBy^G \cap InformedBy^H$.

Note that G and H need to have at least one node or one role in common for their intersection to be non-empty. For example, the intersection of the two graphs in Figures 9(a) and 9(b) yields the OPM graph consisting of the artifacts A and D.

The following is readily verified:

Proposition 5.4. *The intersection of two legal OPM graphs is legal.*

One may wonder about the relations between union and intersection of legal OPM graphs and their temporal theories. We answer this question next. Let G and H now be two legal OPM graphs.

Proposition 5.5. $\mathrm{Th}(G \cup H) = \mathrm{Th}(G) \cup \mathrm{Th}(H)$.

Proof. Each inequality in $\mathrm{Th}(G)$ or $\mathrm{Th}(H)$ corresponds to a single node, a single edge or some use–generate–derive triangle present in G or H. Thus all inequalities present in $\mathrm{Th}(G) \cup \mathrm{Th}(H)$, also belong to $\mathrm{Th}(G \cup H)$. Moreover, the only additional inequalities in $\mathrm{Th}(G \cup H)$ would correspond to some use–generate-derive triangles that were newly formed by the union of G and H. Since both G and H are legal, this is impossible, because legal OPM graphs cannot contain parts of a use–generate-derive triangle. Therefore, $\mathrm{Th}(G \cup H)$ contains only inequalities that are already present in $\mathrm{Th}(G)$, or in $\mathrm{Th}(H)$, or in both. □

Proposition 5.6. $\mathrm{Th}(G \cap H) \subseteq \mathrm{Th}(G) \cap \mathrm{Th}(H)$.

Proof. Any inequality from $\mathrm{Th}(G \cap H)$ corresponds to a single node, an edge, or a use–generate–derive triangle present in $G \cap H$, and thus in both G and H. Therefore, it also belongs to $\mathrm{Th}(G) \cap \mathrm{Th}(H)$. □

The converse inclusion does not hold. If G consists only of edge $P \to A$, and H consists only of edge $A \xrightarrow{!} P$, then $G \cap H$ consists of the two nodes A and P. So

$$\mathrm{Th}(G) = \{\mathrm{create}(A) \preceq \mathrm{end}(P), \mathrm{begin}(P) \preceq \mathrm{end}(P)\} ,$$
$$\mathrm{Th}(H) = \{\mathrm{begin}(P) \preceq \mathrm{create}(A), \mathrm{create}(A) \preceq \mathrm{end}(P), \mathrm{begin}(P) \preceq \mathrm{end}(P)\} ,$$

and

$$\mathrm{Th}(G \cap H) = \{\mathrm{begin}(P) \preceq \mathrm{end}(P)\} .$$

Clearly $\mathrm{create}(A) \preceq \mathrm{end}(P) \in \mathrm{Th}(G) \cap \mathrm{Th}(H) \not\subseteq \mathrm{Th}(G \cap H)$. Note that $\mathrm{create}(A) \preceq \mathrm{end}(P)$ is not even a logical consequence of $\mathrm{Th}(G \cap H)$.

5.1 Renaming and merging

By definition, the nodes and roles of an OPM graph are local to the graph. Prior to performing a union or an intersection of two OPM graphs G and H, we may need to resolve some identity issues between the nodes and roles in the graphs. For example, some process node P in G may represent the same actual process as some process node Q in H. Likewise, role r in edge $P \xrightarrow{r} B$ in G may refer to the same actual role as role s in edge $Q \xrightarrow{s} C$ in H, and also B and C may represent the same actual artifact. Moreover, it is equally possible that some node or role is accidentally used in both graphs whereas this node or role does *not* represent the same actual entity across the two graphs. Resolving such identity issues leads to a renaming operation on one or both of the graphs, whereby nodes and roles representing the same actual entity can be renamed to a common node or role; likewise, nodes and roles not representing the same actual entity, but accidentally used in both graphs, can be renamed to distinct nodes or roles.

Definition 5.7 (Renaming)**.** Let G and H be OPM graphs, which need not be legal. Let ρ_{Art} be a bijection from Art^G to a finite set Art', let ρ_{Proc} be a bijection from $Proc^G$ to a finite set $Proc'$, and let ρ_{Roles} be a bijection from $Roles^G$ to a finite set $Roles'$, with the sets Art', $Proc'$, and $Roles'$ mutually disjoint. Then H is the *renaming* of G by ρ_{Art}, ρ_{Proc}, and ρ_{Roles}, if the following holds:

- $Art^H = Art'$,
- $Proc^H = Proc'$,
- $Roles^H = Roles'$,
- $GeneratedBy!^H = \{(\rho_{Art}(A), \rho_{Roles}(r), \rho_{Proc}(P)) \mid (A, r, P) \in GeneratedBy!^G\}$;
- $Used!^H = \left\{(\rho_{Proc}(P), \rho_{Roles}(r), \rho_{Art}(A)) \mid (P, r, A) \in Used!^G\right\}$;
- $DerivedFrom!^H = \{(\rho_{Art}(A), \rho_{Roles}(r), \rho_{Art}(B)) \mid (A, r, B) \in DerivedFrom!^G\}$;
- $GeneratedBy^H = \left\{(\rho_{Art}(A), \rho_{Proc}(P)) \mid (A, P) \in GeneratedBy^G\right\}$;
- $Used^H = \left\{(\rho_{Proc}(P), \rho_{Art}(A)) \mid (P, A) \in Used^G\right\}$;
- $DerivedFrom^H = \left\{(\rho_{Art}(A), \rho_{Art}(B)) \mid (A, B) \in DerivedFrom^G\right\}$;
- $InformedBy^H = \left\{(\rho_{Proc}(P), \rho_{Proc}(Q)) \mid (P, Q) \in InformedBy^G\right\}$;

Note that Art^G and Art' need not be disjoint; similarly, neither $Proc^G$ and $Proc'$, nor $Roles^G$ and $Roles'$, need to be disjoint. Indeed, ρ_{Art}, ρ_{Proc} and ρ_{Roles} may coincide with the identity function on some of their inputs, i.e., not all artifacts, processes and roles need to be renamed.

Example 5.8. We can rename the graph presented in Figure 9(b) by the following bijections:

- $\rho_{Art}(A) = A$, $\rho_{Art}(C) = B$, $\rho_{Art}(D) = F$, and $\rho_{Art}(E) = E$;
- $\rho_{Proc}(Q) = P$;
- $\rho_{Roles}(s) = r$ and $\rho_{Roles}(s') = r'$.

Then we can take the union of the renamed graph with the graph shown in Figure 9(a), which yields the legal OPM graph presented in Figure 9(c). □

The following is readily verified:

Proposition 5.9. *The renaming of a legal OPM graph is legal.*

We next define the following generalization of renaming.

Definition 5.10 (Merge-renaming)**.** Let G and H be OPM graphs, which need not be legal. Let ρ_{Art}, ρ_{Proc} and ρ_{Roles} be as in Definition 5.7 except that ρ_{Art}, ρ_{Proc} and ρ_{Roles} need not be bijective: they only need to be surjective (onto) mappings. Then we say that H is the *merge-renaming* of G by ρ_{Art}, ρ_{Proc}, and ρ_{Roles}, exactly if the same equalities of Definition 5.7 hold.

Merge-renaming allows the coalescing of two or more nodes to a single node (or two or more roles to a single role). Coalescing of nodes or roles may be performed when analyzing an OPM graph on a coarser level of detail. But coalescing may also be practical when more information becomes available. For example, in a traffic accident scenario, there may be observations about a "blue

car" and other observations about a "Toyota", only to realize later that the blue car is the Toyota.

In contrast to Proposition 5.9, the merge-renaming of a legal OPM graph need not be legal. For example, in Figure 10(c), if we coalesce C and D into a single artifact E, but do not coalesce P and Q, nor their roles, then E has two distinct precise generated-by edges.

As a merge-renaming can coalesce artifacts, such an operation can introduce cycles of derived-from edges to an OPM graph. In the next section, we investigate the consequences of such cycles in OPM graphs.

5.2 Equality inference

Our definitions allow the presence of derived-from cycles in legal OPM graphs. By a *derived-from cycle*, we mean a directed simple cycle composed of derived-from edges (precise or imprecise). An OPM graph resulting from a typical experimental provenance collection procedure does not contain such cycles, and indeed the current OPM reference specification forbids them. For example, it would be strange to assert that A is derived from B and that B is derived from A.

Nevertheless, cycles may arise in a graph when, after a merge operation, certain nodes coalesce. Suppose, for example, that we have three artifacts $A \to B \to C$ without a cycle. If an application does not need the full level of detail provided, it may consider, for example, A and C to be the same at a coarser level of detail. As a consequence, a cycle $A \to B \to C = A$ is created.

Thus, we do not want to disallow derived-from cycles in OPM graphs from the outset. It is important, however, to understand the consequences of the presence of such cycles. We observe that they enforce the equality of certain temporal variables. In the preceding example, we would have $\text{create}(A) = \text{create}(B) = \text{create}(C)$.

First of all, we point out that every OPM graph has a trivial model τ_{triv} consisting of a single time-point t_0 with $\tau_{\text{triv}}(u) = t_0$ for every temporal variable u. Indeed, since the temporal theory of an OPM graph consists only of non-strict inequalities, this interpretation trivially satisfies all non-strict inequalities. Of course, that does not mean that this trivial model is the *only* model that the OPM graph possesses. On the contrary: intuitively, on a fine enough temporal granularity, we should expect that every OPM graph indeed possesses a model where all temporal variables are assigned *distinct* time-points. We observe that this is indeed always possible provided there are no derived-from cycles.

Formally, we fix some OPM graph G for this section. Let us say that a temporal interpretation $(T, \leq, \tau)$ of G is *all-distinct* if $\tau(u) \neq \tau(v)$ for any two distinct temporal variables u and v of G. When, in addition, $\leq$ is a total order on T, we say that τ has the *strict linear order* property.

Proposition 5.11. *If G does not contain any derived-from cycles, then G has an all-distinct temporal model that even satisfies the strict-linear-order property.*

Proof. We construct a total order on all temporal variables of G that is a temporal model of G under the identity mapping. Since G does not contain any derived-from cycles, we can linearly order all artifacts so that no artifact has a derived-from edge from an artifact coming later in the order. Note that there may be many possibilities for such an ordering. Any such ordering imposes an

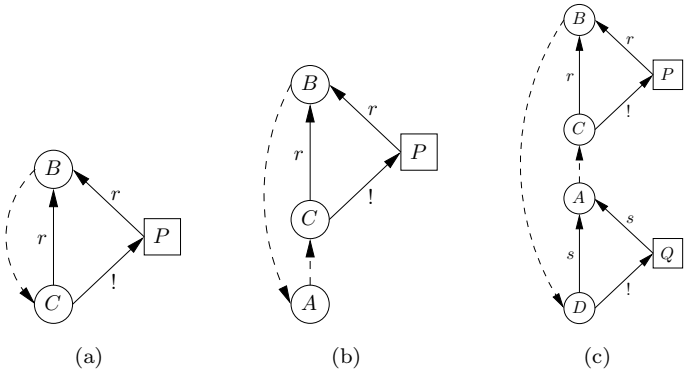

Figure 10: Graph patterns for Proposition 5.13.

ordering on the corresponding create-variables. All begin-variables are placed before all create-variables, in some arbitrary order among them, and similarly all end-variables are placed after all create-variables. Finally, a use-variable involving artifact A is placed immediately as a successor of create(A). If there are more than one use-variables for the same artifact A, they can all be placed in an arbitrary order right after create(A). By inspecting the axioms we see that this order satisfies all axioms. □

We note that Proposition 5.11 does not state that all temporal models of cycle-free graphs are all-distinct. Rather, it establishes that one such all-distinct model exists. The next proposition provides a partial converse to Proposition 5.11:

Proposition 5.12. *If G does contain a derived-from cycle of length at least two, then G cannot have an all-distinct temporal model.*

Proof. Consider a derived-from cycle and let A and B be two distinct artifacts on that cycle. Then any temporal model τ should satisfy $\tau(\text{create}(A)) \leq \tau(\text{create}(B))$ as well as $\tau(\text{create}(B)) \leq \tau(\text{create}(A))$, so $\tau(\text{create}(A))$ equals $\tau(\text{create}(B))$. Hence τ is not all-distinct. □

Propositions 5.11–5.12 do not specify what happens when there are only derived-from cycles of length one (self-loops). Moreover, for a temporal model τ, they do not specify which distinct variables u and v *cannot* have distinct $\tau(u)$ and $\tau(v)$ if the graph contains derived-from cycles. The next proposition fills these gaps by characterizing exactly when two temporal variables must be equal in all temporal models.

Naturally, for two distinct temporal variables u and v of G, we write $\text{Th}(G) \models u = v$ to denote that both $\text{Th}(G) \models u \preceq v$ and $\text{Th}(G) \models v \preceq u$. Thus, if $\text{Th}(G) \models u = v$, then there is no model of G that is all-distinct since any temporal model τ must satisfy $\tau(u) \leq \tau(v)$ and $\tau(v) \leq \tau(u)$; so $\tau(u) = \tau(v)$. Intuitively, two temporal variables u and v of G such that $\text{Th}(G) \models u = v$ can be seen as indistinguishable in the given temporal model, for example as a result of coalescing some nodes in a graph with a more detailed temporal model.

Proposition 5.13. $\text{Th}(G) \models u = v$ *if and only if u and v match one of the following possibilities:*

(1) u is create(A)*, v is* create(B)*, and A and B lie together on a derived-from cycle.*

(2) u is create(B)*, v is* use(P, r, B)*, and in G, the nodes P and B, together with some node C, match the pattern shown in Figure 10(a). Note that B and C need not be distinct.*

(3) u is create(A)*, v is* use(P, r, B)*, and in G, the nodes P, A and B, together with some node C, match the pattern shown in Figure 10(b). Note that A, B and C need not be distinct.*

(4) u is use(P, r, B)*, v is* use(Q, s, A)*, and in G, nodes P, Q, A and B, together with some nodes C and D, match the pattern shown in Figure 10(c). Note that A, B, C and D need not be distinct, nor P and Q.*[6]

Proof. The proof of the if-direction amounts to an inspection of involved patterns to verify that $\mathrm{Th}(G) \models u = v$ indeed holds. For example, let us examine pattern in Figure 10(c) in case 4. By Axiom 3 we have $\mathrm{Th}(G) \models \mathrm{create}(B) \preceq \mathrm{use}(P, r, B)$ for edge $P \xrightarrow{r} B$ and $\mathrm{Th}(G) \models \mathrm{create}(A) \preceq \mathrm{use}(Q, s, A)$ for edge $Q \xrightarrow{s} A$. For the triangle (C, B, P, r) and $G \vdash B \dashrightarrow C$, we can apply Rule 7 from Figure 7, so we have $\mathrm{Th}(G) \models \mathrm{use}(P, r, B) \preceq \mathrm{create}(B)$, and thus $\mathrm{Th}(G) \models \mathrm{create}(B) = \mathrm{use}(P, r, B)$. Likewise, for the triangle (D, A, Q, s) and $G \vdash A \dashrightarrow D$, we obtain $\mathrm{Th}(G) \models \mathrm{create}(A) = \mathrm{use}(Q, s, A)$. Since A and B lie on a derived-from cycle, we also have $\mathrm{Th}(G) \models \mathrm{create}(A) = \mathrm{create}(B)$, hence $\mathrm{Th}(G) \models \mathrm{use}(P, r, B) = \mathrm{use}(Q, s, A)$.

The proof of the only-if direction amounts to a lengthy but straightforward inspection of the possible cases where both $\mathrm{Th}(G) \models u \preceq v$ and $\mathrm{Th}(G) \models v \preceq u$, in the characterization of temporal inference provided by Figure 7. For example, in case 4, we clearly see that we can only obtain $\mathrm{Th}(G) \models \mathrm{use}(P, r, B) = \mathrm{use}(Q, s, A)$ by combining the following rules of Figure 7: Rule 9a with again Rule 9a resulting in the pattern from Figure 10(c) with $A = C$ and $B = D$; Rule 9a with Rule 9b resulting in the pattern from Figure 10(c) with $A = C$; and Rule 9b with again Rule 9b resulting in the pattern from Figure 10(c). □

The patterns of Figures 10(a), 10(b), and 10(c) are respectively super graphs of Axiom 8, and cases 7 and 9B in Figure 7. It is interesting to note that a graph that matches Figure 10(c) also matches Figure 10(b) since $B \dashrightarrow A$ can be inferred from $B \dashrightarrow D$ and $D \xrightarrow{s} A$ in Figure 10(c). Likewise, a graph that matches Figure 10(b) also matches Figure 10(a) since $B \dashrightarrow C$ can be inferred from $B \dashrightarrow A$ and $A \dashrightarrow C$ in Figure 10(b).

Hence, by repeated application of Proposition 5.13 (1)–(4), we derive that all use and create time-points for artifacts A, B, C, D in Figure 10(c) are equal. Likewise, we note the equality of all use and create time-points for artifacts A, B, C in Figure 10(b) and for B, C in Figure 10(a).

The OPM reference specification does not allow derived-from cycles, but it does not define a merge operation either. This section has demonstrated that a merge operation can introduce derived-from loops into OPM graphs, but they must satisfy some constraints: all time-points of artifacts involved in a loop must coalesce to a single time-point. Were a merge operation added to the reference

[6] Note that in Figure 10(c), if C and D coincide then $C \xrightarrow{!} P$ and $D \xrightarrow{!} Q$ must also coincide for G to be legal.

specification, two options are possible. On the one hand, the absence of derived-from cycles can remain a legality constraint, but, therefore, the merge operation should be such that it coalesces all artifacts (and related process) in a loop and removes all self-loops to ensure legality is preserved. On the other hand, the constraint on derived-from cycles can be lifted, as it is in this paper, but the OPM edges decorated with time information and involved in loops should have their time information updated, to reflect the constraints of Proposition 5.13.

6 Refinement and Completion

The OPM reference specification introduces the notion of *refinement* as a relation between two graphs: this relation expresses that one graph represents a more complete description of execution than another graph. The term refinement is inspired by the concept of specification refinement in formal methods [32]. The concept was only intuitively defined as follows: a graph is a refinement of another if dependencies that can be inferred in the original graph are "preserved" in the refinement. The purpose of this section is to formally ground such a notion of refinement in the context of our temporal semantics (Note 11.10).

We fix two legal OPM graphs G and H for use in this section. We also define the following convenient notion.

Definition 6.1. The *logical closure* of a set of inequalities Σ, denoted by $\overline{\Sigma}$, is the set of logical consequences of Σ, i.e., $\overline{\Sigma} = \{\varphi \mid \Sigma \models \varphi\}$.

When context allows, we abbreviate logical closure to closure.

First, we define restriction of an arbitrary set of inequalities Σ to a subset of variables occurring in Σ.

Definition 6.2. Let Σ be a set of inequalities over a set of variables V. Let W be a subset of V. The *restriction* of Σ to W, denoted by $\Sigma|_W$, is the set

$$\Sigma|_W = \{u \preceq v \in \Sigma \mid u, v \in W\} \ .$$

Given our temporal semantics, the intuition of a refinement is the following. Graph H is a refinement of G if all the temporal constraints that can be inferred from $\mathrm{Th}(G)$ can also be inferred from $\mathrm{Th}(H)$. However, such definition is too broad, since refinements can replace nodes by others (say, when a process is implemented by composing two other processes); some temporal constraints of $\mathrm{Th}(G)$ may range over temporal variables that do not exist in $\mathrm{Th}(H)$. Hence, H is a refinement of G, if the temporal constraints that can be inferred from $\mathrm{Th}(G)$ over the common set of variables between H and G, can also be inferred from $\mathrm{Th}(H)$. Formally, the definition is expressed as follows.

Definition 6.3 (Refinement)**.** H is a *refinement* of G if

$$\overline{\mathrm{Th}(G)}|_{Vars(G) \cap Vars(H)} \subseteq \overline{\mathrm{Th}(H)} \, .$$

In this definition, it is not necessary to restrict $\overline{\mathrm{Th}(H)}$ because $\overline{\mathrm{Th}(H)}$ is on the right-hand side of a set containment. Indeed, for any three sets A, B and C, we have that $(A \cap B) \subseteq C$ iff $(A \cap B) \subseteq (C \cap B)$.

Note that Theorem 4.4 can be effectively used to decide whether a given graph H is a refinement of a given graph G. Still, the definition of refinement is strictly semantic and does not provide much guidance towards constructing a refinement. An interesting open problem is to find a finite set of graph operations that all result in refinements, and such that every refinement can be obtained by using these operations.

6.1 Graph operations and refinement

In this section we investigate the relations between the graph operations defined in Section 5 and refinement. We first investigate the subgraph operation, the union and the intersection. Let G and H be two legal OPM graphs.

Proposition 6.4. *If H is a legal subgraph of G, then G is a refinement of H.*

Proof. Since H is a subgraph of G, it is clear from Definition 3.5 that $\mathrm{Th}(H) \subseteq \mathrm{Th}(G)$. Hence $\overline{\mathrm{Th}(H)}|_{Vars(H)\cap Vars(G)} = \overline{\mathrm{Th}(H)} \subseteq \overline{\mathrm{Th}(G)}$ as desired. □

Note that a legal subgraph of G is not necessarily a refinement of G. For instance, let G be a use–generate–derive triangle (A, B, P, r), and let H be a subgraph of G consisting of $P \xrightarrow{r} B$ and $A \xrightarrow{!} P$. Then H is legal, but is not a refinement of G: $\mathrm{create}(B) \preceq \mathrm{create}(A) \in \overline{\mathrm{Th}(G)}|_{Vars(G)\cap Vars(H)}$ yet $\mathrm{create}(B) \preceq \mathrm{create}(A) \notin \overline{\mathrm{Th}(H)}$.

The above proposition also applies to the union and intersection, and their operands.

Corollary 6.5. *If $G \cup H$ is legal, then $G \cup H$ is a refinement of G.*

Note that G is not necessarily a refinement of a legal $G \cup H$. For example, let G consist of $A \to B$ and node P and let H consist of $A \xrightarrow{!} P$ and node B. So $G \cup H$ is legal and consists of $A \to B$ and $A \xrightarrow{!} P$. Then G is not a refinement of $G \cup H$: $G \cup H \vdash P \to B$, so $\mathrm{create}(B) \preceq \mathrm{end}(P) \in \overline{\mathrm{Th}(G \cup H)}|_{Vars(G\cup H)\cap Vars(G)}$ yet $\mathrm{create}(B) \preceq \mathrm{end}(P) \notin \overline{\mathrm{Th}(G)}$.

However, if G and H are node-disjoint, then G is a refinement of $G \cup H$.

Corollary 6.6. *G is a refinement of $G \cap H$.*

Note that $G \cap H$ is not necessarily a refinement of G. For example, let G consists of $A \to Q$, $A \xrightarrow{!} P$ and $P \to Q$, and let H consist of $A \to Q$ and $P \to Q$. Then $G \cap H$ equals H, which is not a refinement of G: $\mathrm{create}(A) \preceq \mathrm{end}(P) \in \overline{\mathrm{Th}(G)}|_{Vars(G)\cap Vars(H)}$, yet $\mathrm{create}(A) \preceq \mathrm{end}(P) \notin \overline{\mathrm{Th}(G \cap H)}$.

Next we investigate the merge-renaming operation. Let H be a renaming of G by ρ_{Art}, ρ_{Proc} and ρ_{Roles} (conforming to Definition 5.7). Then G is not necessarily a refinement of H, or vice versa. For instance, let G consist of $A \to B$, let $\rho_{Art}(A) = B$ and $\rho_{Art}(B) = A$, then H consists of $B \to A$. They are clearly not each others refinements.

Still, if one needs to apply a merge-renaming on a graph before performing a union or an intersection, a merge-renaming operation that preserves the temporal constraints of the original would be advisable. This motivates the following restricted version of the merge-renaming operation:

Definition 6.7 (Proper merge-renaming)**.** Let ρ_{Art}, ρ_{Proc} and ρ_{Roles} be as in Definition 5.10, and let ρ be the point-wise union of ρ_{Art}, ρ_{Proc} and ρ_{Roles}. Then we use $\rho(G)$ to denote the merge-renaming of G by ρ_{Art}, ρ_{Proc} and ρ_{Roles}.

We call a merge-renaming $\rho(G)$ *proper* if for any node (or role) x in G, if $\rho(x) \neq x$ but $\rho(x)$ is also a node (or role) in G, then $\rho(\rho(x)) = \rho(x)$.

Intuitively, in a proper merge-renaming operation, some nodes/roles are preserved, whereas all the others are either renamed into new ones or coalesced into preserved ones. Such an operation disallows arbitrary renaming and permutation of nodes/roles. Although it seems overly restrictive, it makes sense in the OPM model, because nodes and roles are actually *identifiers*. Hence, coalescing of nodes/roles, or renaming them into new ones, is a form of identity resolution. For example, witnesses of a car accident may speak of a "blue car" or a "Toyota". Later on, one realizes the witnesses talked about the same car, so both "blue car" and "Toyota" should be renamed to the car's registration number (while "blue car" and "Toyota" become its annotations). Likewise, hospitals frequently admit unconscious patients under a temporary ID. Later on, they are either matched to an already known patient or to a new ID if the patient is admitted for the first time.

We conclude this section by showing that a proper and legal merge-renaming of a graph is indeed a refinement of the original graph.

Theorem 6.8. *Let G be a legal OPM graph, and let $\rho(G)$ be a proper and legal merge-renaming of G, for some ρ. Then $\rho(G)$ is a refinement of G.*

To prove the theorem we use the following auxiliary lemmas.

Lemma 6.9. *Let G be a legal OPM graph, and let $\rho(G)$ be a legal merge-renaming of G, for some ρ. Then if $G \vdash X \dashrightarrow Y$, also $\rho(G) \vdash \rho(X) \dashrightarrow \rho(Y)$, i.e., a legal merge-renaming preserves edge-inference.*

The above lemma is readily verified.

Lemma 6.10. *Let G be a legal OPM graph, and let $\rho(G)$ be a proper merge-renaming of G, for some ρ. If a node (or role) x belongs to both G and to the image of ρ, then $\rho(x) = x$.*

Indeed, if x is in the image of ρ, then there exist a node (role) y in G, such that $\rho(x) = y$. If $x = y$, then $\rho(x) = x$ holds immediately. If $x \neq y$, then $\rho(y) \neq y$, so $\rho(\rho(y)) = \rho(y)$ (by Definition 6.7). Thus $\rho(x) = x$ holds as desired.

We can now prove the theorem. We need to prove the following:

$$\overline{\mathrm{Th}(G)}|_{Vars(G) \cap Vars(\rho(G))} \subseteq \overline{\mathrm{Th}(\rho(G))}\,.$$

Let $Commons = Vars(G) \cap Vars(\rho(G))$. For each inequality $\varphi \in \overline{\mathrm{Th}(G)}|_{Commons}$, we must show that φ belongs to $\overline{\mathrm{Th}(\rho(G))}$. We know from Theorem 4.4, illustrated in Figure 7, that each such inequality is associated with a pattern in G. Therefore, we need to find a similar pattern in $\rho(G)$ that produces *exactly* the same inequality.

Let $\varphi \in \overline{\mathrm{Th}(G)}|_{Commons}$. We follow the axioms and rules presented in Figure 7, the axioms first, to cover all possible forms that φ may assume:

(a) φ is $\text{begin}(P) \preceq \text{end}(P)$, for some P in G. Since $\text{begin}(P), \text{end}(P) \in \underline{\mathit{Commons}}$, P is also present in $\rho(G)$. By Axiom 1, $\text{begin}(P) \preceq \text{end}(P) \in \overline{\text{Th}(\rho(G))}$.

(b) φ is $\text{begin}(P) \preceq \text{create}(A)$ or $\text{create}(A) \preceq \text{end}(P)$, for some $A \xrightarrow{!} P$ in G. Since $\text{create}(A), \text{begin}(P), \text{end}(P) \in \mathit{Commons}$, we also have A and P in $\rho(G)$. From $A \xrightarrow{!} P$ in G, by Definition 6.7, we have $\rho(A) \xrightarrow{!} \rho(P)$ in $\rho(G)$. We can apply Lemma 6.10 to A and P, obtaining $\rho(A) = A$ and $\rho(P) = P$. Hence, $A \xrightarrow{!} P$ is also in $\rho(G)$ and, by Axiom 2, both $\text{begin}(P) \preceq \text{create}(A)$ and $\text{create}(A) \preceq \text{end}(P)$ belong to $\overline{\text{Th}(\rho(G))}$. Note that $A \xrightarrow{!} P$ in G and $A \xrightarrow{!} P$ in $\rho(G)$ need not have the same role.

(c) φ is one of the following: $\text{begin}(P) \preceq \text{use}(P, r, A)$, $\text{use}(P, r, A) \preceq \text{end}(P)$, or $\text{create}(A) \preceq \text{use}(P, r, A)$, for some $P \xrightarrow{r} A$ in G. The variable $\text{use}(P, r, A)$ belongs to $\mathit{Commons}$; that is only possible if edge $P \xrightarrow{r} A$ is also present in $\rho(G)$. By Axiom 3, $\text{begin}(P) \preceq \underline{\text{use}(P, r, A)}$, $\text{use}(P, r, A) \preceq \text{end}(P)$, and $\text{create}(A) \preceq \text{use}(P, r, A)$ belong to $\overline{\text{Th}(\rho(G))}$.

(d) φ is $\text{use}(P, r, B) \preceq \text{create}(A)$, for some $G \triangle (A, B, P, r)$. For $P \xrightarrow{r} B$ in G we apply case c, so $P \xrightarrow{r} B$ is also in $\rho(G)$. For $A \xrightarrow{!} P$ in G, by case b, $A \xrightarrow{!} P$ belongs to $\rho(G)$. We can apply Lemma 6.10 to A, B, and r, obtaining $\rho(A) = A$, $\rho(B) = B$, and $\rho(r) = r$. For $A \xrightarrow{r} B$ in G, we apply Definition 6.7, so $\rho(A) \xrightarrow{\rho(r)} \rho(B)$ in $\rho(G)$, and thus $A \xrightarrow{r} \underline{B \text{ in } \rho(G)}$. Clearly, $\rho(G) \triangle (A, B, P, r)$, hence $\text{use}(P, r, B) \preceq \text{create}(A) \in \overline{\text{Th}(\rho(G))}$ (Axiom 8).

(e) φ is $\text{create}(B) \preceq \text{create}(A)$, for $G \vdash A \dashrightarrow B$. Since $\text{create}(A), \text{create}(B) \in \mathit{Commons}$, A and B also belong to $\rho(G)$. From $G \vdash A \dashrightarrow B$, by Lemma 6.9, we have $\rho(G) \vdash \rho(A) \dashrightarrow \rho(B)$. We can apply Lemma 6.10 to A and B, obtaining $\rho(A) = A$ and $\rho(B) = B$. Hence $\underline{\rho(G) \vdash A \dashrightarrow B}$ as desired and, by Rule 1, $\text{create}(B) \preceq \text{create}(A) \in \overline{\text{Th}(\rho(G))}$.

(f) φ is $\text{begin}(P) \preceq \text{create}(A)$, for $G \vdash A \dashrightarrow P$. Since $\text{create}(A), \text{begin}(P), \text{end}(P) \in \mathit{Commons}$, A and P belong to $\rho(G)$. From $G \vdash A \dashrightarrow P$, by Lemma 6.9, $\rho(G) \vdash \rho(A) \dashrightarrow \rho(P)$. We can apply Lemma 6.10 to A and P, obtaining $\rho(A) = A$ and $\underline{\rho(P) = P}$. Thus $\rho(G) \vdash A \dashrightarrow P$ and, by Rule 2, $\text{begin}(P) \preceq \text{create}(A) \in \overline{\text{Th}(\rho(G))}$.

(g) φ is $\text{create}(A) \preceq \text{end}(P)$, for $G \vdash P \dashrightarrow A$. Since $\text{create}(A), \text{begin}(P), \text{end}(P) \in \mathit{Commons}$, A and P belong to $\rho(G)$. From $G \vdash P \dashrightarrow A$, by Lemma 6.9, we have $\rho(G) \vdash \rho(P) \dashrightarrow \rho(A)$. We can apply Lemma 6.10 to A and P, obtaining $\rho(A) = A$ and $\rho(P) = P$. Therefore, $\rho(G) \vdash P \dashrightarrow A$ and, by Rule 3, $\text{create}(A) \preceq \text{end}(P) \in \overline{\text{Th}(\rho(G))}$.

(h) φ is $\text{begin}(Q) \preceq \text{end}(P)$, for $G \vdash P \dashrightarrow Q$. Since $\text{begin}(P), \text{end}(P), \text{begin}(Q), \text{end}(Q) \in \mathit{Commons}$, P and Q belong to $\rho(G)$. From $G \vdash P \dashrightarrow Q$, by Lemma 6.9, $\rho(G) \vdash \rho(P) \dashrightarrow \rho(Q)$. We can apply Lemma 6.10 to P and Q, obtaining $\rho(P) = P$ and $\underline{\rho(Q) = Q}$. Hence $\rho(G) \vdash P \dashrightarrow Q$ and, by Rule 4, $\text{begin}(Q) \preceq \text{end}(P) \in \overline{\text{Th}(\rho(G))}$.

(i) φ is $\text{create}(B) \preceq \text{use}(P, r, A)$, for $P \xrightarrow{r} A$ in G and $G \vdash A \dashrightarrow B$. By applying cases c and e to $P \xrightarrow{r} A$ in G and $G \vdash A \dashrightarrow B$, respectively,

we have $P \xrightarrow{r} A$ in $\rho(G)$ and $\rho(G) \vdash A \dashrightarrow B$. By Rule 5, create$(B) \preceq$ use$(P, r, A) \in \overline{\text{Th}(\rho(G))}$.

(j) φ is begin$(Q) \preceq$ use(P, r, A), for $P \xrightarrow{r} A$ in G and $G \vdash A \dashrightarrow Q$. By applying cases c and f to $P \xrightarrow{r} A$ in G and $G \vdash A \dashrightarrow Q$, respectively, we have $P \xrightarrow{r} A$ in $\rho(G)$ and $\rho(G) \vdash A \dashrightarrow Q$. By Rule 6, begin$(Q) \preceq$ use$(P, r, A) \in \overline{\text{Th}(\rho(G))}$.

(k) φ is use$(P, r, C) \preceq$ create(A), for $G \triangle (B, C, P, r)$ and $G \vdash A \dashrightarrow B$. By applying cases d and e to $G \triangle (B, C, P, r)$ and $G \vdash A \dashrightarrow B$, respectively, we have $\rho(G) \triangle (B, C, P, r)$ and $\rho(G) \vdash A \dashrightarrow B$. By Rule 7, use$(P, r, C) \preceq$ create$(A) \in \overline{\text{Th}(\rho(G))}$.

(l) φ is use$(P, r, B) \preceq$ end(Q), for $G \triangle (A, B, P, r)$ and $G \vdash Q \dashrightarrow A$. By applying cases d and g to $G \triangle (A, B, P, r)$ and $G \vdash Q \dashrightarrow A$, respectively, we have $\rho(G) \triangle (A, B, P, r)$ and $\rho(G) \vdash Q \dashrightarrow A$. By Rule 8, use$(P, r, B) \preceq$ end$(Q) \in \overline{\text{Th}(\rho(G))}$.

(m) φ is use$(P, r, B) \preceq$ use(Q, s, A), for $G \triangle (C, B, P, r)$ and $Q \xrightarrow{s} A$ in G, and either $A = C$ or $G \vdash A \dashrightarrow C$. By applying cases d and c to $G \triangle (C, B, P, r)$ and $Q \xrightarrow{s} A$, respectively, we obtain $\rho(G) \triangle (C, B, P, r)$ and $Q \xrightarrow{s} A$ in $\rho(G)$. If $A = C$, then A clearly belongs to G and to the image of ρ, so, by Lemma 6.10, $\rho(A) = A$. If $G \vdash A \dashrightarrow C$, then, by case e, $\rho(G) \vdash A \dashrightarrow C$. We can thus apply either Rule 9a or Rule 9b, so use$(P, r, B) \preceq$ use$(Q, s, A) \in \overline{\text{Th}(\rho(G))}$.

6.2 Completion Operations

The edge-inference rules introduced in Definition 4.3 allow new edges to be inferred in an OPM graph, the nodes of the graph remaining unchanged. The OPM reference specification defines *completion operations*, which add new nodes and new edges to a graph.

The rationale for such completion operations was to provide syntactic transformations over graphs, which offer an *explanation* for some OPM edges. First, we present the operations graphically and intuitively, before formalizing them. According to Figure 11(a), *process introduction* states that if an artifact B was derived from an artifact A, then there exist a process R and role r such that B was generated by R with role r and R used A. The operation does not specify which process was involved, nor the role of B with regard to this process, but it states that such a process R and role r existed. Likewise, *artifact introduction*, presented in Figure 11(b), states that if there is a process Q that was informed by a process P, then there exist an artifact C and a role s such that Q used C precisely with role s and C was generated by P. Again, the completion does not specify which artifact and role were involved. (Note 11.11)

We can see that process introduction generalizes the use–generate–derive triangle of Figure 2 to imprecise derived-from and used-edges. We refer to this triangle as an *imprecise use–generate–derive triangle*. Likewise, artifact introduction recognizes the existence of a complementary *imprecise use–generate–inform triangle*. (We note that there is no precise use–generate–inform triangle, since informed-by edges are always imprecise.)

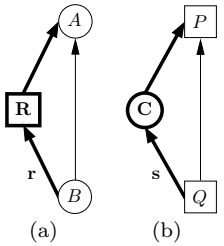

Figure 11: Completion operations: (a) process introduction and (b) artifact introduction.

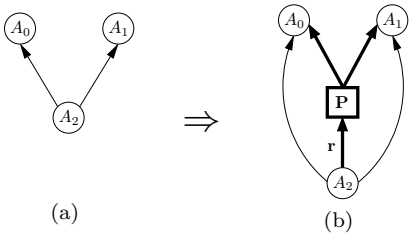

Figure 12: Process introduction producing a legal graph.

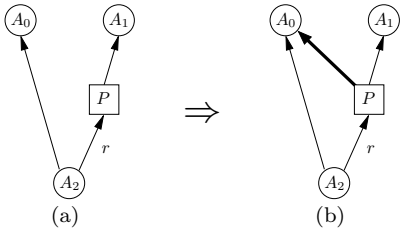

Figure 13: When completion becomes edge inference.

Completion operations should be considered in conjunction with legality constraints. For instance, Figure 12(a) depicts an artifact A_2 derived from two artifacts A_0 and A_1. The process introduction operation can be applied twice here, but the legality constraint requires an artifact to be generated by a single process. Hence, Figure 12(b) displays the *only possible* completion, where introduced process P used both A_0 and A_1, and generated A_2.

Application of process insertion to the graph of Figure 13(a) entails that there is a process that used A_0 and that generated A_2 precisely, but the legality constraint implies that this process is P. Hence, we can derive that P used A_0, which is the inference of a used-edge as in Definition 4.3.

Completion operations can introduce new nodes in a graph, but can result in some uncertainty as illustrated by Figures 14 and 15. In Figure 14, it is unknown whether the two processes P_0 and P_1 introduced by the process introduction operation are identical. Likewise, in Figure 15, it is not known whether the two artifacts A_0 and A_1 introduced by artifact introduction are the same. This uncertainty is formalized below to the effect that completion is a nondeterministic operation: a given graph may have several possible completions.

To define completion formally, we first formalize the notions of triangle relevant to completion operations.

Definition 6.11 (Complete Triangle for $B \to A$)**.** Let G be an OPM graph with two artifacts A and B and an imprecise edge $B \to A$. Graph G contains a complete triangle for $B \to A$ if there exists a process R and role r in G, with edges $B \xrightarrow{r} R$ and $R \to A$. We then say that B, A, R, r constitute an imprecise use–generate–derive triangle.

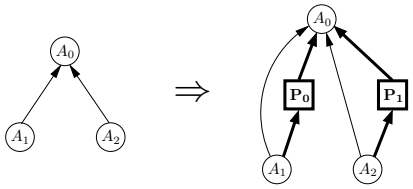

Figure 14: Uncertainty: is $P_0 = P_1$ or $P_0 \neq P_1$?

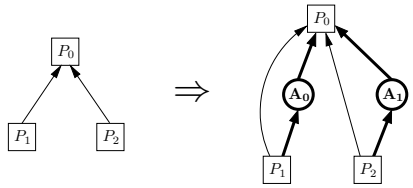

Figure 15: Uncertainty: is $A_0 = A_1$ or $A_0 \neq A_1$?

Definition 6.12 (Complete Triangle for $Q \to P$)**.** Let G be an OPM graph with two processes Q and P and an informed-by edge $Q \to P$. Graph G contains a complete triangle for $Q \to P$ if there exists an artifact C and role s in G, with edges $Q \xrightarrow{s} C$ and $C \to P$. We then say that Q, P, C, s constitute an imprecise use–generate–inform triangle.

There can be at most one complete triangle for a given imprecise edge $B \to A$ in a legal OPM graph G, since B can only be generated by one process R. On the other hand, there may be several complete triangles for a given imprecise edge $Q \to P$: for instance, two artifacts C_1, C_2 could be generated by P and used by Q, with respective roles s_1 and s_2.

We then define a completion operation as possibly introducing a node, a role, and edges, as appropriate, in order to form complete triangles.

Definition 6.13 (CompletionOperation)**.** Let G be an OPM graph. A graph H results from a completion operation on G, if H was obtained as follows:

- H is the result of *process introduction* for $B \to A$ in G, with $A, B \in Art^G$, if there exists a process R and a role r, such that:
  - $Proc^H = Proc^G \cup \{R\}$,
  - $Roles^H = Roles^G \cup \{r\}$,
  - $GeneratedBy!^H = GeneratedBy!^G \cup \{(B, r, R)\}$,
  - $Used^H = Used^G \cup \{(R, A)\}$,
  - all other sets remain the same;
- H is a result of *artifact introduction* for $Q \to P$ in G, with $P, Q \in Proc^G$, if there exists an artifact C and a role s, such that:
  - $Art^H = Art^G \cup \{C\}$,
  - $Roles^H = Roles^G \cup \{s\}$,
  - $GeneratedBy^H = GeneratedBy^G \cup \{(C, P)\}$,
  - $Used!^H = Used!^G \cup \{(Q, s, C)\}$,
  - all other sets remain the same.

We note that a completion operation can introduce new nodes R and C or reuse existing ones in G. Likewise, they can create new edges or reuse existing ones. In some cases, a completion operation is exactly an edge inference (cf. inference of used-edges in Definition 4.3).

Not all completion operations lead to legal graphs. Hence, a completion operation applied to legal graph G is said to be *valid* if it results in a legal graph H. We then call H a *valid completion* of G.

Since G is always a subgraph of H, the result of a completion is always a refinement of G (Proposition 6.4). The converse does not hold, however. Indeed, let us consider a graph G with imprecise edges $C \rightarrow A$ and $C \rightarrow B$, and an isolated process P. If graph H is the result of completing $C \rightarrow A$ via P, we can infer $H \vdash P \dashrightarrow B$. Hence, the inequality create$(B) \preceq$ end(P) holds in H but not in G. Alternatively, let us consider a graph G with imprecise edges $Q \rightarrow P$ and $B \rightarrow A$. If graph H is the completion of $Q \rightarrow P$ via B, then $H \vdash Q \dashrightarrow A$, which cannot be inferred in G.

So, in summary, graphs can be completed non-deterministically, but completions are a refinement of the original graph.

7 Multi-account OPM graph

In the OPM reference specification [20], OPM graphs can contain multiple *accounts*. Accounts are used to identify parts of a large, integrated, OPM graph, in which each account is a coherent OPM graph in itself. An account should be perceived as one perspective on what happened during a past execution, as illustrated by Alice's and Bob's account in Section 2.

We define a multi-account OPM graph as follows:

Definition 7.1 (Multi-account OPM graph)**.** A multi-account OPM graph is a structure

$$(\mathit{Art}, \mathit{Proc}, \mathit{Roles}, \mathit{GeneratedBy}!, \mathit{Used}!, \mathit{DerivedFrom}!, \\ \mathit{GeneratedBy}, \mathit{Used}, \mathit{DerivedFrom}, \mathit{InformedBy}, \mathit{accountOf})$$

where

- *Art* and *Proc* are two disjoint finite sets of elements called *artifacts* and *processes*, respectively;
- *Roles* is a finite set of elements called *roles*;
- $\mathit{GeneratedBy}! \subseteq \mathit{Art} \times \mathit{Roles} \times \mathit{Proc}$;
- $\mathit{Used}! \subseteq \mathit{Proc} \times \mathit{Roles} \times \mathit{Art}$;
- $\mathit{DerivedFrom}! \subseteq \mathit{Art} \times \mathit{Roles} \times \mathit{Art}$;
- $\mathit{GeneratedBy} \subseteq \mathit{Art} \times \mathit{Proc}$;
- $\mathit{Used} \subseteq \mathit{Proc} \times \mathit{Art}$;
- $\mathit{DerivedFrom} \subseteq \mathit{Art} \times \mathit{Art}$;
- $\mathit{InformedBy} \subseteq \mathit{Proc} \times \mathit{Proc}$;
- *accountOf* is a function from $\mathit{Nodes} \cup \mathit{Edges}$ to $\mathbb{P}^{\mathrm{fin}}(\mathit{Account})$, the set of all finite subsets of *Account*, where
  - *Account* is a finite set of elements called *accounts*,

– $Nodes = Art \cup Proc$, and
– $Edges = GeneratedBy! \cup Used! \cup DerivedFrom! \cup GeneratedBy \cup Used \cup DerivedFrom \cup InformedBy$.

The sets Art, $Proc$, $Roles$ and $Account$ are mutually disjoint.

In a multi-account OPM graph G, every node and edge of G can be associated with zero, one, or more accounts, which is captured by the function $accountOf$, provided as the last constituent of graph G.

For a given OPM graph G, for any $A \in Art$, we call $accountOf(A)$ the *account membership* of A in G. Likewise, for any $P \in Proc$, we call $accountOf(P)$ the account membership of P in G.

For a precise edge e in G, of the form (x, r, y), or for an imprecise edge e in G, of the form (x, y), we say that x and y are *incident* to e, and denote this by the predicates $isIncident(x, e)$ and $isIncident(y, e)$.

To ensure that the source and destination of an edge belong to the same account as the edge, a node "inherits" and cumulates the account memberships of the edges it is incident to. Formally, this is expressed as follows.

Definition 7.2 (Effective Account)**.** For a given OPM graph G we define the *effective-account function*

$$effectiveAccountOf^G : Nodes \cup Edges \to \mathbb{P}^{\text{fin}}(Account)$$

as follows:

- If $x \in Nodes$, then

$$effectiveAccountOf^G(x) = accountOf(x) \cup \bigcup \{accountOf(e) \mid e \in Edges \text{ and } isIncident(x, e)\}.$$

- If $x \in Edges$, then

$$effectiveAccountOf^G(x) = accountOf(x).$$

An *account view* is the single-account OPM graph extracted from a multi-account OPM graph by restricting attention to the nodes and edges associated with a given account. Formally, this operation is defined as follows.

Definition 7.3 (Account View)**.** For a given multi-account OPM graph G and an account α, we define the *account view of G according to α*, denoted by $view(G, \alpha)$, as follows:

- $Art^{view(G,\alpha)} = \left\{A \in Art \mid \alpha \in effectiveAccountOf^G(A)\right\}$;
- $Proc^{view(G,\alpha)} = \left\{P \in Proc \mid \alpha \in effectiveAccountOf^G(P)\right\}$;
- $Roles^{view(G,\alpha)} = \{r \in Roles \mid (x, r, y) \text{ is an edge in } view(G, \alpha)\}$;
- $GeneratedBy!^{view(G,\alpha)} = \left\{(A, r, P) \mid \alpha \in effectiveAccountOf^G(A, r, P)\right\}$;

- $Used!^{view(G,\alpha)} = \Big\{(P, r, A) \mid \alpha \in effectiveAccountOf^G(P, r, A)\Big\}$;
- $DerivedFrom!^{view(G,\alpha)} = \Big\{(A, r, B) \mid \alpha \in effectiveAccountOf^G(A, r, B)\Big\}$;
- $GeneratedBy^{view(G,\alpha)} = \Big\{(A, P) \mid \alpha \in effectiveAccountOf^G(A, P)\Big\}$;
- $Used^{view(G,\alpha)} = \Big\{(P, A) \mid \alpha \in effectiveAccountOf^G(P, A)\Big\}$;
- $DerivedFrom^{view(G,\alpha)} = \Big\{(A, B) \mid \alpha \in effectiveAccountOf^G(A, B)\Big\}$;
- $InformedBy^{view(G,\alpha)} = \Big\{(P, Q) \mid \alpha \in effectiveAccountOf^G(P, Q)\Big\}$.

Definition 7.4. A multi-account OPM graph is called *legal* if all its account views are legal.

Note that in a legal multi-account OPM graph, we can perform temporal inferences on each account view, although separately. It is difficult to define temporal inference on the whole graph, since each account potentially provides a different perspective. However, one can perform various graph operations to combine different accounts into a single one, and then perform temporal inference on the latter. One can also establish whether one account is a refinement of another one.

8 Related work

With well over 400 publications [19] on the topic of provenance, this section focuses on OPM-specific related work. For a broader perspective, we refer the reader to comprehensive surveys for provenance and e-science [27], provenance and databases [1, 3] and provenance and the Web [19].

Cheney [2] investigates the use of structural causal models as a semantics for provenance graphs, and relates some OPM concepts to notions of actual cause and explanation proposed by Halpern and Pearl [9] (Note 11.12). The semantics that we propose here is similar to Cheney's in the sense that it provides a mathematical meaning for OPM graphs, however, it differs in other significant ways: *(i)* our semantics conforms to the OPM reference specification [20] and in particular handles time, all permitted OPM edges, accounts, algebraic operations and refinements; Cheney's semantics is account-less and regards single-step derivation edges as inferable, when they can only be asserted in OPM, and does not characterize inferred edges; *(ii)* our semantics is purely temporal, and does not see an OPM graph as a function operating on some inputs and generating outputs; this view of OPM graphs is complementary to ours, but relies on meanings associated with processes by means of annotations, which have not been modeled here. *(iii)* Cheney's semantics attempts the more ambitious goal of providing a global approximation (using the predictive nature of causal models) for the program being executed (without having its explicit code), so that its behavior can be repeated for any arbitrary input; our semantics is silent about the predictive nature of OPM graphs.

Missier and Goble [17] address the question of whether, for any OPM graph, there exists a plausible workflow in the Taverna workflow language, which could

have generated the graph. To this end, they identify the extra information that should be captured as part of an OPM graph so that the mapping from OPM to a workflow representation can be derived. Thus, this work derives an executable semantics for OPM, obtained by composing their translation and the Taverna semantics. It however does not tackle OPM in full, ignoring accounts, time and refinement; their translation should be revisited to leverage the distinction between precise and imprecise edges, introduced in this paper. Similarly, Kwasnikowska and Van den Bussche [10] map the NRC data flow model, a formally specified data model for workflows, to OPM.

Moreau [18] proposes the reproducibility semantics for OPM, which is a denotational semantics characterizing how an OPM graph can be used to reproduce a past computation; a "re-execution" of such a graph results in a new OPM graph, and mapping of nodes from the original graph to the new graph. By doing so, he identifies a class of reproducible OPM graphs. The reproducibility semantics assumes a mapping of each process to a function (taking some inputs artifacts, and generating some output artifact), and, like Cheney's causal semantics of OPM, sees an OPM graph as a function operating on inputs and producing outputs. Moreau defines a notion of refinement, corresponding to the nested execution of procedures. Future work could try to integrate Moreau's reproducibility semantics and the temporal semantics presented in this paper.

As part of the W3C Provenance Incubator activity [31], mappings of multiple provenance ontology's to OPM were defined [26]. These mappings showed that concepts such as processes and artifacts mapped quite naturally between models. The mappings did not take into account the temporal meaning of the various data models; revisiting these mappings in the light of this temporal semantics would provide a better correspondence between ontology's. Likewise, Miles explains how Dublin Core provenance-related concepts can be translated into OPM graphs [15]. Given that Dublin Core also introduces time, a finer-grained mapping could be derived, based on the temporal semantics introduced in this paper.

Several approaches are specializing OPM to specific application domains or facets of computing. Groth and Moreau introduce the D-Profile [8], as a specialization of OPM for distributed systems. Their profile comprises artifact and process types and graph patterns to describe communications in distributed systems. Similarly, Freitas et al. introduce types of processes, artifacts and agents to describe data publication over the Web [6].

Several teams have adopted OPM and implemented inferences, as prescribed by the reference specification. To this end, many teams have exploited Semantic Web technologies, such as OWL and SWRL, to implement OPM reasoning: e.g., Tupelo [14], OPMO [22], Provenance Challenge 3 Tetherless [5], or, reproducibility service [18]; alternatively, some teams used recursive queries in relational databases: e.g., OPMProv [12]. To the best of our knowledge, none of these teams support use time-points and patterns introduced in this paper. In this respect, the inferences they make are incomplete.

9 Future work

According to the OPM reference specification, a process is controlled by an agent, and this control dependency can be assigned a starting and an ending

time. This paper provides a temporal semantics for a subset of OPM that does not include agents. The reason why agents were not incorporated in this semantics is that this notion is not clearly defined, and novel proposals are beginning to emerge [23]. The absence of agents has some implication for this temporal semantics: the starting time-point (resp. ending time-point) of a process is not preceded (resp. followed) by any other time-point in a graph temporal theory. Future work would be to extend our temporal semantics to OPM agents.

Our semantics has introduced the notion of precise edge, which stipulates that an artifact was used (or generated) within the scope of a process, i.e. after its starting time, and before its ending time. Imprecise edges are silent about the position of their time-points with respect to a process beginning and ending times. Other precise edges could be considered, such as a used-edge known to precede the beginning of a process, or a generated-by edge known to follow the ending of a process. Such edges would allow novel inferences to be made, which would need to be characterized as well.

Adopting Lamport's definition of parallel events [11], two variables u, v in $Vars(G)$ could be defined as parallel events if neither $u \preceq v$ nor $v \preceq u$ are logical consequences of G's theory. In future work, one could take a parallel perspective on OPM, and investigate patterns of parallelism in OPM graphs.

This paper, in Section 6, is the first to provide a formal definition of refinements in OPM. We see this definition as a starting point for novel research directions. As already mentioned, it would be useful to explore the existence of a finite set of graph transformations that could be used to derive all possible refinements. Also, the notion of refinement could take further constraints into consideration, such as process nesting (as considered by Moreau [18]).

10 Conclusion

The Open Provenance Model is a data model for provenance that is issued from requirements from a community of practitioners involved in the Provenance Challenge activity. This data model, the temporal constraints it identifies, and the inferences it permits have all been defined informally. As Cheney et al. [4] argue this may lead to difficulty in understanding what provenance actually represents, and what kind of reasoning can legitimately be performed with it. From a practical viewpoint, definitions that are not precisely formulated may lead to systems that do not inter-operate, which is particularly problematic for a language aiming at inter-operability. The problem is highlighted by the misunderstandings and guesses that occurred in the mapping activity between provenance vocabularies [26].

To address this problem, this paper proposes a temporal semantics for the OPM data model. The originality of the approach is to leverage the temporal constraints introduced by OPM and make them a core constituent of the meaning of OPM graphs: a temporal theory of an OPM graph consists of a set of such constraints. From this temporal theory, notions of interpretation, model and logical consequence can be established, within a purely semantic context, which provides us with a reference framework to characterize inferences of OPM edges. An important contribution of this paper is to identify that OPM inferences are sound with respect to the temporal theory, but not all logical

Table 4: Table for Note 11.4.

OPM reference specification	This paper
`Used`	*Used*! and *Used*
`WasGeneratedBy`	*GeneratedBy*! and *GeneratedBy*
`WasDerivedFrom`	*DerivedFrom*! and *DerivedFrom*
`Used`* (asserted)	*Used*
`Used`* (inferred)	$\dashrightarrow$
`WasGeneratedBy`* (asserted)	*GeneratedBy*
`WasGeneratedBy`* (inferred)	$\dashrightarrow$
`WasDerivedFrom`* (asserted)	*DerivedFrom*
`WasDerivedFrom`* (inferred)	$\dashrightarrow$
`WasTriggeredBy` (asserted)	*InformedBy*
`WasTriggeredBy` (inferred)	$\dashrightarrow$
`WasControlledBy`	n/a

consequences can be derived by means of inferences over OPM graphs. The paper however identifies a set of graph patterns, which allow a completeness result to be established. Exploiting this novel semantic framework, the paper then defines graph operations (such as union and intersection), graph relations (such as refinement) and accounts in terms of the temporal semantics.

11 OPM-specific notes

Note 11.1. The OPM reference specification also includes agents, which are entities controlling processes. We do not formalize such a concept here, since it is the subject of multiple ongoing discussions. However, in future work section, we discuss ways of introducing them in the model, and their potential implications on temporal models.

Note 11.2. The OPM reference specification also includes edges of the type WasControlledBy (from a process to an agent) which are not included in this paper, because we do not model agents. Furthermore, we prefer to adopt the term *informed-by* rather than *was-triggered-by* since its informal meaning is more aligned to its formal definition.

Note 11.3. The OPM reference specification does not associate a role with a was-derived-from edge. It is a contribution of this paper to have identified the need of this role for defining a temporal semantics of OPM.

Note 11.4. Table 4 maps sets in the OPM reference specification to sets or notations introduced in this paper.

Note 11.5. The OPM reference specification has a different legality definition. The first legality requirement of Definition 3.2 is exactly the same as in [20], but the second requirement was not stated before. Indeed, we have only discovered during the work reported in this paper that the second requirement is needed in order to give a clean semantics to precise derived-from edges. Finally, the OPM specification has an additional condition that we do not need here: there may

be no cycles in the was-derived-from edges. We discuss further the implications of cycles in derived-from edges in OPM graphs in Section 5.2.

Note 11.6. OPM introduces an observation interval that we do not formalize here. Our only assumption about time-points is that they can be ordered partially.

Note 11.7. The OPM reference specification associates the create time-point with edges of the type `WasGeneratedBy`, since different accounts can assign different time-points to a same edge. In the presence of an edge of the type `WasControlledBy`, beginning and ending times of a process are associated with the edge.

Note 11.8. The OPM reference specification defines a used-edge as meaning that the process required the availability of the artifact to be able to complete its execution: this is exactly Axiom 6. Likewise, the reference specification defines a was-generated-by edge as meaning that the process was required to initiate its execution for the artifact to have been generated: this is exactly Axiom 5. The OPM reference specification defines a was-triggered-by edge exactly as Axiom 7.

In addition, OPM time constraints indicate that the use time-point follows the start of the process, and the generate time-point precedes the end of the process. The OPM reference specification is silent about the time constraints related to multi-step edges of the types `Used*` and `WasGeneratedBy*`. It is a contribution of this paper to have integrated all these constraints into a single set of axioms.

Note 11.9. The OPM reference specification is not specific about the difference between the precise was-derived-from edge $A \xrightarrow{!} B$ and its imprecise version $A \rightarrow B$. First, a precise edge $A \xrightarrow{!} B$ can only exist in the presence of a generate–use–derive triangle, according to Definition 3.2. Second, both the create and use time-points (for A and B, respectively) occur within the scope of the process identified by the triangle. These constraints do not hold for $A \rightarrow B$.

Note 11.10. Technically, the OPM reference specification defines refinement between two accounts. However, like graph operations, it is useful to define refinement over OPM graphs. They naturally transpose to accounts introduced in Section 7.

Note 11.11. The OPM reference specification defines three completion operations: process and artifact introduction, as in this paper, but also artifact elimination. Artifact elimination is precisely the inference of *informed-by* in Definition 4.3.

Note 11.12. The OPM reference specification defines edges as *causal* relationships, but does not provide an explanation for this terminology, and the kind of causality underpinning them. In this paper, we have chosen not to adopt this terminology to avoid unnecessary technical jargon. However, our temporal semantics provides a clarification with regard to this notion. Indeed, the kind of causality underlying OPM relationships is the causality typically defined in distributed systems. The constraints $u \preceq v$ in our temporal theory are similar in nature to Tel's causal order [28, Definition 2.20]. In particular, according to Tel, a send event precedes the corresponding receive event in a distributed system, very much like our create time-point precedes a use time-point for the same artifact. Similar orders were referred to as "causality relation" by Mattern [13] and "happened before" by Lamport [11].

References

[1] Peter Buneman, James Cheney, Wang-Chiew Tan, and Stijn Vansummeren. Curated databases. In *PODS '08: Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–12, New York, NY, USA, 2008. ACM.

[2] J. Cheney. Causality and the semantics of provenance. In S.B. Cooper, E. Kashefi, and P. Panangaden, editors, *Proceedings 6th Workshop on Developments in Computational Models*, volume 26 of *EPTCS*, pages 63–74, 2010.

[3] James Cheney, Laura Chiticarius, and Wang-Chiew Tan. Provenance in Databases: Why, How, and Where. *Foundations and Trends in Databases*, 1(4):379–474, 2009.

[4] James Cheney, Stephen Chong, Nate Foster, Margo Seltzer, and Stijn Vansummeren. Provenance: A Future History. In *Companion to the 24th Annual ACM SIGPLAN Conference on Object-Oriented Programming Languages, Systems, Languages, and Applications: Onward! Session*, pages 957–964, 2009.

[5] Li Ding, James Michaelis, Jim McCusker, and Deborah L. McGuinness. Linked provenance data: A semantic Web-based approach to interoperable workflow traces. *Future Generation Computer Systems*, 2010. In Press, http://dx.doi.org/10.1016/j.future.2010.10.011.

[6] Andre Freitas, Sean O'Riain, Edward Curry, and Tomas Knap. W3P: Building an OPM based provenance model for the Web. *Future Generation Computer Systems*, 2010. In Press, http://dx.doi.org/10.1016/j.future.2010.10.010.

[7] Yolanda Gil, Ewa Deelman, Mark Ellisman, Thomas Fahringer, Geoffrey Fox, Dennis Gannon, Carole Goble, Miron Livny, Luc Moreau, and Jim Myers. Examining the challenges of scientific workflows. *IEEE Computer*, 40(12):26–34, 2007.

[8] Paul Groth and Luc Moreau. Representing Distributed Systems Using OPM. *Future Generation Computer Systems*, December 2010. In Press, http://dx.doi.org/10.1016/j.future.2010.10.001.

[9] Joseph Y. Halpern and Judea Pearl. Causes and Explanations: A Structural-Model Approach. Part I: Causes. *Br J Philos Sci*, 56(4):843–887, 2005.

[10] Natalia Kwasnikowska and Jan Van den Bussche. Mapping the NRC Dataflow Model to the Open Provenance Model. In Juliana Freire, David Koop, and Luc Moreau, editors, *Second International Provenance and Annotation Workshop, IPAW'2008*, volume 5272 of *Lecture Notes in Computer Science*, pages 3–16. Springer, 2008.

[11] Leslie Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(7):558–565, 1978.

[12] Chunhyeok Lim, Shiyong Lu, Artem Chebotko, and Farshad Fotouhi. Storing, Reasoning, and Querying OPM-Compliant Scientific Workflow Provenance Using Relational Databases. *Future Generation Computer Systems*, 2010. In Press, http://dx.doi.org/10.1016/j.future.2010.10.013.

[13] Friedemann Mattern. Virtual time and global states of distributed systems. In M. Cosnard et al., editors, *Proceedings of the International Workshop on Parallel and Distributed Algorithms*, pages 215–226, Amsterdam, 1989. Elsevier Science Publishers.

[14] Robert E. McGrath and Joe Futrelle. Reasoning about provenance with OWL and SWRL rules. In *AAAI Spring Symposium 2008 "AI Meets Business Rules and Process Management"*, 2008.

[15] Simon Miles. Mapping Attribution Metadata to the Open Provenance Model. *Future Generation Computer Systems*, 2010. In Press, http://dx.doi.org/10.1016/j.future.2010.10.007.

[16] Simon Miles, Paul Groth, Miguel Branco, and Luc Moreau. The requirements of using provenance in e-Science experiments. *Journal of Grid Computing*, 5(1):1–25, 2007.

[17] Paolo Missier and Carole Goble. Workflows to Open Provenance Graphs, round-trip. *Future Generation Computer Systems*, 2010. In Press, http://dx.doi.org/10.1016/j.future.2010.10.012.

[18] Luc Moreau. Provenance-Based Reproducibility in the Semantic Web. Technical report, University of Southampton, 2010. http://eprints.ecs.soton.ac.uk/21554/.

[19] Luc Moreau. The Foundations for Provenance on the Web. *Foundations and Trends in Web Science*, 2(2–3):99–241, November 2010.

[20] Luc Moreau, Ben Clifford, Juliana Freire, Joe Futrelle, Yolanda Gil, Paul Groth, Natalia Kwasnikowska, Simon Miles, Paolo Missier, Jim Myers, Beth Plale, Yogesh Simmhan, Eric Stephan, and Jan Van den Bussche. The Open Provenance Model core specification (v1.1). *Future Generation Computer Systems*, 2010. In Press, http://dx.doi.org/10.1016/j.future.2010.07.005.

[21] Luc Moreau, Bertram Ludaescher, Ilkay Altintas, Roger S. Barga, Shawn Bowers, Steven Callahan, George Chin Jr., Ben Clifford, Shirley Cohen, Sarah Cohen-Boulakia, Susan Davidson, Ewa Deelman, Luciano Digiampietri, Ian Foster, Juliana Freire, James Frew, Joe Futrelle, Tara Gibson, Yolanda Gil, Carole Goble, Jennifer Golbeck, Paul Groth, David A. Holland, Sheng Jiang, Jihie Kim, David Koop, Ales Krenek, Timothy McPhillips, Gaurang Mehta, Simon Miles, Dominic Metzger, Steve Munroe, Jim Myers, Beth Plale, Norbert Podhorszki, Varun Ratnakar, Emanuele Santos, Carlos Scheidegger, Karen Schuchardt, Margo Seltzer, Yogesh L. Simmhan, Claudio Silva, Peter Slaughter, Eric Stephan, Robert Stevens, Daniele Turi, Huy Vo, Mike Wilde, Jun Zhao, and Yong Zhao. The first provenance challenge. *Concurrency and Computation: Practice and Experience*, 20(5):409–418, 2008.

[22] Luc Moreau (editor), Li Ding, Joe Futrelle, Daniel Garijo Verdejo, Paul Groth, Mike Jewell, Simon Miles, Paolo Missier, Jeff Pan, and Jun Zhao. Open Provenance Model (OPM) OWL Specification. Technical report, openprovenance.org, 2010. http://openprovenance.org/model/opmo.

[23] James Myers. I Think Therefore I Am Someone Else: Understanding the confusion of granularity with Continuant/Occurrent and Related Perspective Shifts. In *Provenance and Annotation of Data and Processes*, volume 6378 of *Lecture Notes in Computer Science*, pages 292–294. Springer Berlin / Heidelberg, 2010.

[24] PREMIS Working Group. Data Dictionary for Preservation Metadata — Final Report of the PREMIS Working Group. Technical report, Preservation Metadata: Implementation Strategies (PREMIS), 2005. http://www.oclc.org/research/projects/pmwg/premis-final.pdf.

[25] Christoph Ringelstein and Steffen Staab. Papel: A language and model for provenance-aware policy definition and execution. In Richard Hull, Jan Mendling, and Stefan Tai, editors, *Business Process Management*, volume 6336 of *Lecture Notes in Computer Science*, pages 195–210. Springer Berlin / Heidelberg, 2010.

[26] Satya Sahoo, Paul Groth, Olaf Hartig, Simon Miles, Sam Coppens, James Myers, Yolanda Gil, Luc Moreau, Jun Zhao, Michael Panzer, and Daniel Garijo. Provenance Vocabulary Mappings. Technical report, W3C, 2010. http://www.w3.org/2005/Incubator/prov/wiki/Provenance_Vocabulary_Mappings.

[27] Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-Science. *SIGMOD Record*, 34(3):31–36, 2005.

[28] Gerard Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, 1994.

[29] The Provenance Challenge Wiki. http://twiki.ipaw.info/bin/view/Challenge/.

[30] J.D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume II. Computer Science Press, 1989.

[31] W3C Incubator Activity, Provenance Incubator Group Charter. http://www.w3.org/2005/Incubator/prov/charter.

[32] Jim Woodcock and Jim Davies. *Using Z. Specification, Refinement, and Proof*. Prentice Hall, 1996.

[33] Jun Zhao. The Open Provenance Model Vocabulary Specification. Technical report, University of Oxford, 2010. http://purl.org/net/opmv/ns.