COMPUTING SCIENCE

Provenance graph abstraction by node grouping

Paolo Missier, Jeremy Bryans, Carl Gamble, Vasa Curcin and Roxana Danger

TECHNICAL REPORT SERIES

No. CS-TR-1393 August 2013

TECHNICAL REPORT SERIES

No. CS-TR-1393 August, 2013

Provenance graph abstraction by node grouping

P. Missier, J. Bryans, C. Gamble, V. Curcin and R. Danger

Abstract

Data provenance is a form of metadata that records the activities involved in data production. It can be used to help data consumers to form judgments regarding data reliability. The PROV data model, released by the W3C in 2013, defines a relational model and constraints which provides a structural and semantic foundation for provenance. This enables the exchange of provenance between data producers and consumers. When the provenance content is sensitive and subject to disclosure restrictions, however, a complementary model is needed to enable producers to partially obfuscate provenance in a principled way. In this paper we propose such a formal model. It is embodied by a grouping operator, whereby a set of nodes in a PROV-compliant provenance graph is replaced by a new abstract node, leading to a new valid PROV graph. We define graph editing rules which allow existing dependencies to be removed, but guarantee that no spurious dependencies are introduced in the abstracted graph. As grouping is closed with respect to composition, it can be used as a building block to achieve complex abstraction. The operator is implemented as part of a user tool that lets owners of sensitive provenance information specify custom abstraction policies.

Printed and published by Newcastle University,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

MISSIER, P., BRYANS, J., GAMBLE, C., CURCIN, V., DANGER, R.

Provenance graph abstraction by node grouping
[By] Paolo Missier, Jeremy Bryans, Carl Gamble, Vasa Curcin, Roxana Danger

Newcastle upon Tyne: Newcastle University: Computing Science, 2013.

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1393)

Added entries

NEWCASTLE UNIVERSITY
Computing Science. Technical Report Series. CS-TR-1393

Abstract

Data provenance is a form of metadata that records the activities involved in data production. It can be used to help data consumers to form judgments regarding data reliability. The PROV data model, released by the W3C in 2013, defines a relational model and constraints which provides a structural and semantic foundation for provenance. This enables the exchange of provenance between data producers and consumers. When the provenance content is sensitive and subject to disclosure restrictions, however, a complementary model is needed to enable producers to partially obfuscate provenance in a principled way. In this paper we propose such a formal model. It is embodied by a grouping operator, whereby a set of nodes in a PROV-compliant provenance graph is replaced by a new abstract node, leading to a new valid PROV graph. We define graph editing rules which allow existing dependencies to be removed, but guarantee that no spurious dependencies are introduced in the abstracted graph. As grouping is closed with respect to composition, it can be used as a building block to achieve complex abstraction. The operator is implemented as part of a user tool that lets owners of sensitive provenance information specify custom abstraction policies.

About the authors

Dr. Paolo Missier is a Lecturer in Information and Knowledge Management with the School of Computing Science, Newcastle University, UK. His current research interests include models and architectures for the management and exploitation of data provenance, specifically extensions of e-infrastructures for scientific provenance. He is the PI of the EPSRC-funded project "Trusted Dynamic Coalitions", investigating provenance based policies for information exchanges amongst partners in the presence of limited trust. Paolo contributes to two Provenance-focused Working Groups: he is a member of the W3C Working Group on Provenance on the Web, where co-edited the Provenance Conceptual Data Model; and he is co-lead of the Provenance Working Group of the NSF-funded DataONE project. He serves on a voluntary basis in both capacities. Paolo holds a Ph.D. in Computer Science from the University of Manchester, UK (2007), a M.Sc. in Computer Science from University of Houston, Tx., USA (1993) and a B.Sc. and M.Sc. in Computer Science from Universita' di Udine, Italy (1990).

Jeremy is a Senior Research Associate in the School of Computing Science, and a member of Centre for Software Reliability. His research interests are in the modelling and analysis of collaborating systems, and in the development of trustworthy policies for their interaction. He is currently a Co-Investigator on the EPSRC project Trusting Dynamic Coalitions, on which he works on designing and composing provenance policies for coalition members. Part of his time is spent on the EU project COMPASS, on which is developing semantic foundations for a modelling language for Systems of systems.

Carl received his BEng (Hons) in Manufacturing Systems and Mechanical Engineering from the University of Leeds in 1996. Then between 1997 and 2003 he worked as an Industrial Engineer for Faurecia in Washington where he had responsibility for developing a synchronous delivery system and the programming of ABB water jet robots. In 2003 Carl left Faurecia and started an MSc in Computing Science at the University of Newcastle, graduating with a Distinction in 2004. This led to him starting a PhD under the supervision of Dr Cristina Gacek investing the design time detection of architectural mismatch between service oriented architecture components. While completing his PhD Carl took on a role as an RA on the Ministry of Defence funded Software Systems Engineering Initiative (SSEI) project. He was involved in the dependability sub task, specifically looking at using dependability metadata to drive software reconfigurations in the field of NEC systems. This resulted in the development of prototype tooling to simulate the operation and reconfiguration of Systems-of-systems in user defined scenarios. Carl is currently working on the DESTECS project. This project, made up of a consortium of industrial and academic partners, is developing tools and methodology for the collaborative development and simulation of embedded systems. The Newcastle team is principally concerned with methodology and how fault-tolerance can be incorporated into these collaborative, multi-disciplinary models.

Dr Roxana Danger is currently a Research Associate at the Computing Department of Imperial College London. She works on provenance APIs and data mining tools for EU-FP7 project TRANSFoRm (http://www.transformproject.eu/). She was previously at the Department of Computer Systems and Computation, Universidad Politecnica de Valencia, Spain. She developed an information extraction system for Protein-Protein interactions which combined Machine Learning and Description Logics. She obtained her PhD at University Jaume I, Castellon, Spain, with a project aimed at extracting and analysing semantic data from archaeology site excavation reports. Previously, she worked at University of Oriente in Santiago de Cuba as a full time professor.

Dr Vasa Curcin (VC) is a Research Fellow in the Social Computing Group and London e-Science Centre in the Department of Computing at Imperial College London. Dr Curcin is the Scientific Manager of the EU FP7 TRANSFoRm project, and the main author of its provenance framework. He is also the technical lead of the NIHR CLAHRC Northwest London project, and the designer of the WISH process improvement toolkit in use by over 600 users in London hospitals. He is one of the principal authors of the E-Science Discovery Net workflow system. His research interests are in the areas of process model transformations from abstract specifications and business models, to executable workflows and associated provenance traces.

Suggested keywords

PROVENANCE METADATA
PROVENANCE ABSTRACTION

Provenance graph abstraction by node grouping

Paolo Missier, Jeremy Bryans, Carl Gamble, Vasa Curcin, and Roxana Danger

Abstract—Data provenance is a form of metadata that records the activities involved in data production. It can be used to help data consumers to form judgments regarding data reliability. The PROV data model, released by the W3C in 2013, defines a relational model and constraints which provides a structural and semantic foundation for provenance. This enables the exchange of provenance between data producers and consumers. When the provenance content is sensitive and subject to disclosure restrictions, however, a complementary model is needed to enable producers to partially obfuscate provenance in a principled way. In this paper we propose such a formal model. It is embodied by a grouping operator, whereby a set of nodes in a PROV-compliant provenance graph is replaced by a new abstract node, leading to a new valid PROV graph. We define graph editing rules which allow existing dependencies to be removed, but guarantee that no spurious dependencies are introduced in the abstracted graph. As grouping is closed with respect to composition, it can be used as a building block to achieve complex abstraction. The operator is implemented as part of a user tool that lets owners of sensitive provenance information specify custom abstraction policies.

Index Terms—Provenance metadata, provenance abstraction

1 Introduction

The provenance of data is a form of structured metadata that records the activities involved in data production. In addition to activities, a provenance trace may include input or intermediate data products, as well as references to agents, humans as well as software systems, who were responsible for carrying out those activities. In multi-party collaborations settings that involve data sharing, as well as in third party auditing of data and processes, there is a broad expectation that shipping the available provenance to collaborators, or more generally publishing it along with the data may help data consumers form judgments regarding the reliability of the data. The recent standardisation of a data model for exchanging provenance in an interoperable way, namely the PROV model from the W3C [MMB+12], makes this assumption realistic. With the emergence of provenance as a valuable complement to data in such a setting, two separate but related issues arise, namely of interpretation of complex provenance, and of selective disclosure of potentially sensitive provenance content.

First, regarding *complexity* we observe that while provenance is a rich form of metadata, it is often the reflection of articulated processes involving many intermediate data products and many inter-related activities. Such complexity is at odds with the need for humans to understand traces, common for instance in e-science, where provenance is perceived as an important element of scholarly communication, and as such it needs to be understood by non-technical users, i.e., scientists. Indeed in e-science, the idea of abstracting provenance to make it understandable is not entirely new. The Zoom system [BBDH08], discussed in more detail below, is an example of a system for computing views over provenance, which is designed to simplify human understanding of provenance traces.

Second, regarding *selective disclosure*, we are motivated by the emerging notion of *dynamic coalitions* [BFJM06]. These are ad hoc collaborative partnerships that are created to pursue a common goal, in scenarios such as multi-agency emergency / threat responses, as well as exchange of intelligence information. Despite the need to share data of a possibly sensitive nature, these coalitions are characterized by a lack of established interaction protocols and by limited trust amongst the partners. Our working assumption is that, in this setting, provenance can be used as a form of evidence in support of the data that is being exchanged. In these scenarios, full disclosure of provenance data may not be possible, e.g. due either to confidentiality policies, Intellectual Property restrictions associated with individual components, or data protection regulations. These restrictions introduce the need to hide certain parts of the provenance traces. In this paper we develop a model and algorithm for performing abstraction over provenance metadata. These provide the theoretical underpinning, defined on top of PROV, for applications that require provenance exchange while obfuscating some of its content.

1.1 Example scenario: provenance of intelligence information

Imagine a setting where one or more public agencies engage in a business relationship with an independent intelligence provider. In Fig. 1 these are the Public Agencies (PA) box and the Incident Room Analysts (IR) box, respectively. IR has a business incentive to provide

- *P. Missier, J. Bryans and C. Gamble are with the School of Computing Science, Newcastle University, UK. E-mail: firstname.lastname@ncl.ac.uk*
- *V. curcin and R. Danger are with the Department of Computing, Imperial College, London, UK*

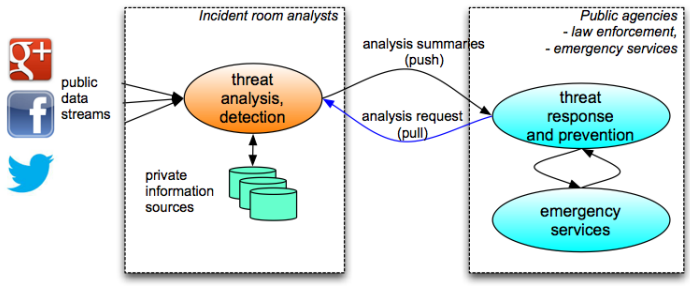

Fig. 1. Simple two-way partnership involving one sender (IR) and one receiver (PA)

analysis of potential threats to PA that is as accurate as possible, while PA has an interest in acquiring and acting upon intelligence reports issued by IR. Realistically, however, PA will want to mitigate the risk of acting upon information provided by IR, which is potentially unreliable. At the same time, IR has a business incentive to supply PA with additional evidence that facilitates PA's risk assessment.

The key assumption that motivates our work is that the provenance of the intelligence report is relevant in contributing, at least in part, the needed evidence. This may contain a wealth of details regarding how the report was produced, including the raw input data used for the analysis, the analytical tools and their configuration parameters, as well as the identity of the analysts involved, and their position within the business organization.

An example of provenance containing such information is shown in Fig. 2. As depicted in the figure, provenance can be represented as a graph whose nodes represent either *entites* (ovals in the figure), i.e., data, documents, etc., *activities* (rectangles), which represent the execution of some process over a period of time, or *entites* (pentagons), which represent humans or computing systems. Directed edges represent various types of relationships, the most common being "activity a used entity e", "entity e was generated by activity a", "activity a was associated with agent ag" (i.e., ag was responsible for a), and more. Provenance graphs are introduced more formally in Sec. 3.

While both IR and PA are interested in sending and receiving such provenance information, some of it may be sensitive and proprietary. Input datasets may include a combination of public social media streams, shown in the figure, and private databases. Likewise, analytical tools may include proprietary algorithms. In a setting where mutual trust is limited, the contrast between the need to exchange provenance as a form of evidence on one side, and the need to protect confidential information that can be found in the provenance on the other, generates a tension amongst the partners.

We believe that such tension can be resolved by providing IR, the *provenance owner*, with ways to control the disclosure of provenance to third parties. This consists of (i) a model of abstraction over provenance, whereby some of its elements are grouped together and replaced with a new, abstract element, and (ii) a policy model that the provenance owner can use to specify the elements which are to be abstracted.

1.2 Contribution: graph abstraction by node grouping

The work presented here is focused exclusively on (i), while (ii) is the subject of a separate upcoming report, focused on a user tool for the specification and enforcement of provenance abstraction policies. For completeness, however, we provide a brief overview of the tool in Sec. 6, and an example of an abstraction policy in the Appendix.

Removal of information from a provenance graph can be achieved in a number of ways. First, one could simply remove the labels as well as the annotations from invidual nodes and relationships, i.e., anonymize part of the graph. Second, nodes and relationships can be removed from the graph. This will reduce the information available, and may even break the graph into two or more unconnected subcomponents. For example, removing nodes `consolidateAJC` and `consolidateBNC` in the graph of Fig. 2 would create two unconnected subcomponents, although the resulting subgraphs are still valid.

Both these options are straightforward and are not discussed further in the paper. Neither, however, reduce the complexity of the graph (arguably, node removal may in fact make the resulting incomplete provenance harder to understand and analyze). The mechanism presented here instead involves a new *grouping* operator, whereby a user-selected set of nodes is replaced by a single abstract node, and connectivity is guaranteed by replacing some of the relationships amongst those nodes with new ones that involve the new abstract node, in a way that guarantees that no false dependencies are introduced.

Our specific contribution in the remainder of the paper is the formal definition of the grouping operator, first with reference to graphs containing only entities and activities, and then extended to graphs with agents. Using examples to justify our formal choices, we define functions that achieve grouping by rewriting the original graph into a new, abstracted graph, and show that the rewriting maps preserves the validity of the original graph, i.e., the validity of the PROV typing constraints.

1.3 Compliance to PROV

We view abstraction as a form of graph rewriting. Conceptually, two approaches can be taken when defining the rewriting rules. One option is to extend the PROV model with additional elements (for instance, new types to denote abstract nodes) and a corresponding system of constraints, resulting in an extension PROV' of PROV. In this setting, a legal rewriting is one that transforms a valid PROV graph PG into a valid PROV'

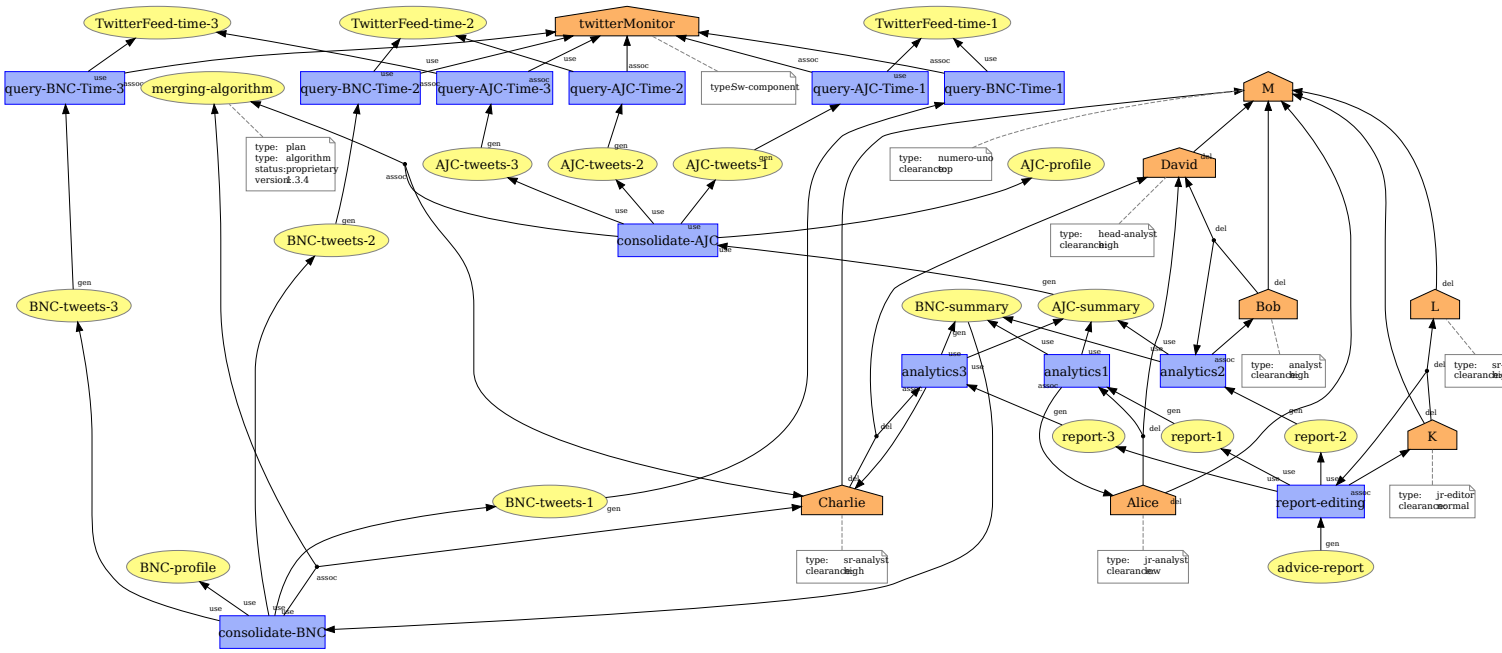

Fig. 2. Example provenance graph for the reference scenario (Fig. 1)

graph PG', where validity is interpreted as conformance to the PROV′ schema. While this approach leads to the definition of a new and possibly interesting model for *abstract provenance*, the interoperability guarantees provided from having standardized PROV would be compromised.

A second option, which we take here, is to define a rewriting that generates a PROV-compliant graph. Should additional metadata describing the abstract nature of the some of elements of the final graph PG' be required — and more generally, the provenance of PG' relative to the transformation — this can be itself expressed using PROV, and associated to PG' using PROV's "provenance of provenance" mechanism (i.e., bundles [MMB+12]). By taking this approach, interoperability is preserved by expressing abstraction as part of PROV itself. Considering the role of provenance as evidence for the process of data production, in addition to defining validity-preserving rewritings we also require that no false or "spurious" relationships be introduced by the rewriting. For instance, if a and e are unrelated in PG, it would be illegal to add a new relationship between them, although this will not violate validity with respect to the schema. Application of this principle requires some care when either of these nodes is replaced by an abstract node, as in this case a new relationship involving the abstract node may be legitimate. These situations are discussed as part of our formalization in Sec. 4.

2 Related Work

2.1 Abstraction over provenance

Work on provenance abstraction generally combines two elements, namely a technique or algorithm for graph editing, and a policy framework to drive the algorithm. As mentioned, in this paper we focus exclusively on the former, while the latter is described in a separate demo paper.

Regarding transforming provenance graphs for the purpose of either reducing their complexity, and/or removing sensitive information, several approaches stand out. The first is the Zoom system [BBDH08], designed to compute *views* over provenance graphs. Its main assumption is that the graph is a trace that specifically represents the execution of a dataflow. This is a common occurrence in e-science, where workflows that follow the dataflow model are a popular high level programming paradigm. In this setting views over provenance are effectively a form of abstraction and are computed based on the user's indication of which workflow modules (tasks) are relevant, or perhaps based on which modules the user has access to. Thus, key to this approach is knowledge of the underlying workflow structure, which is used to specify the nodes in the graphs to be abstracted. This sets Zoom apart from our work, which instead investigates the properties of a grouping operator independently of the origins of the trace to which it is applied.

Also specific to workflow-generated provenance, and thus too narrow in scope for our purposes, is a strand of research that investigates the problem of preserving the privacy of functions used in workflows, when a large number of input/output pairs for those functions is re-

vealed through the provenance traces of multiple workflow executions. This work on *module privacy* [DKR+11], [DKRB10], [DKM+11] is concerned with protecting the semantics of workflow modules. It applies anonymization techniques specifically to provenance graphs and is again centred around a workflow-specific form of provenance and is thus also peripheral to our interest.

Closer to our abstraction model, both in motivation and in its technical approach, is the ProPub system [DZL11], which computes views over provenance graphs that are suitable for publication by meeting certain privacy requirements. In ProPub, users specify edit operations on a graph, such as anonymizing, abstracting, and hiding certain parts of it. The operations are specified as logic rules, and are interpreted natively by the Datalog-based prototype implementation. ProPub adopts an "apply–detect–repair" approach, whereby user rules are applied to the graph first, then consistency violations that may occur in the resulting new graph are detected, and a final set of edits are applied to the graph in order to repair such violations. In some cases, this causes nodes that the user wanted removed to be reintroduced, and it is not always possible to satisfy all rules. In contrast, our grouping involves more simply a set of nodes to be abstracted (but note that anonymization is a particular case, when the group contains a single element). In return for this simplicity in the specification of the nodes to be grouped, our method always produces a valid abstract graph while ensuring that the nodes specified in the policy are removed.

Finally, recent work on *provenance redaction* [CKKT11b] employs a graph grammar technique to edit provenance that is expressed using the Open Provenance Model [MCF+11] (a precursor to PROV), as well as a redaction policy language. Although the authors claim that the redaction operators ensure that specific relationships are preserved, this critical issue is not addressed formally in the paper, i.e., with reference to the OPM semantics. In contrast, the formal schema and set of constraints that come with PROV [MMB+12], [CMM12] provide the necessary grounding for reasoning about the validity-preservation properties of the editing operations.

2.2 General graph anonymization

For completeness, we briefly mention more general techniques for graph editing, largely motivated by the need to preserve privacy in social network data. This body of work, which is not specific to provenance, extends the well-known data anynomization framework developed for relational data to graph data structures [ZG08], [BCKS09], [LT08]. The main idea is to randomly remove arcs between two nodes and replace them with new ones. As arcs in PROV graphs represent relationships with a given semantics, this approach generally results in false dependencies being created in the edited graph, and is therefore not viable. The main value of this body of work in this setting, as summarised in [ZPL08], is to ensure that various forms of anonymization are provably robust to attacks from adversaries who can potentially leverage their partial information about fragments of the graph, to infer additional knowledge. In this paper we do not discuss the robustness of abstraction by grouping, indeed we do not consider any specific threats, and so the challenge of preventing the reconstruction of the abstracted fragments of provenance graphs is left for future work.

2.3 Provenance Access Control

Most of the work on protecting access to sensitive provenance includes policy models that extend traditional data access control frameworks (RBAC), with a distinction made between PBAC (Provenance-Based Access Control) and PAC (Provenance Access Control). PBAC is about policy to specify access rights to data objects based on their provenance. An example, from [NPS12], is a rule of the form "only the student submitter can access the graded homework object". This rule can be enforced by looking for a dependency path in a provenance graph, whereby a given homework is attributed to a specific student (i.e., relation *IsAuthoredBy* in the Open Provenance Model). This assumes that the object's attribution is explicit in the provenance graph. It is less clear how such a rule would be evaluated when the provenance is incomplete with respect to such attribution dependency, however.

PAC, or how to enforce access control on parts of a provenance graph, is more directly relevant to our work. An analysis of some of the challenges associated with secure provenance exchange can be found in [BSS08], where examples are presented that show how the provenance of data can be more sensitive than the data itself. Another position paper [HSW07] describes the challenges associated with the exchange of provenance across multiple partners, in a setting where forgery of provenance by malicious users is a possibility, and where users may collude to reveal sensitive provenance to others. These are all common and complex security problems. Unfortunately, the paper stops short of providing any hints at technical solutions, and indeed it is not clear how these problems are specific to provenance, as opposed to data sharing in general.

A concrete specification of an access control system or provenance [CKKT11a] consists of a XACML-based policy language, in which path queries are used to specify target elements of the graph, as well as an implementation architecture and a prototype.

3 Core PROV model

We now introduce the core elements of the PROV model, which forms the basis for the grouping operator. We maintain a dual view of provenance, both as a relational model (with binary relations) and as a graph model. Viewed as a relational model, PROV includes three types

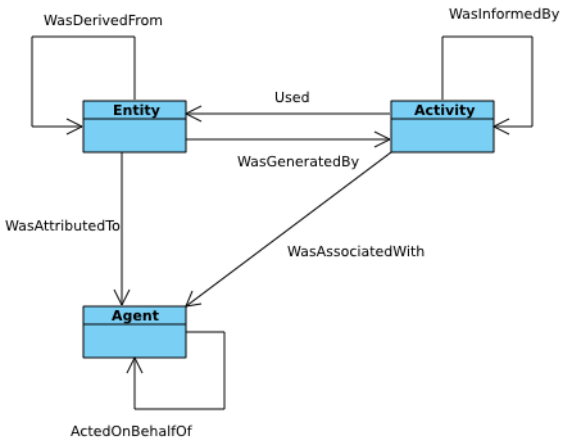

Fig. 3. Core elements of the PROV model, from [MMB+12]

of entities: Entities (En), Activities (Act), and Agents (Ag), and several types of relations amongst them. In line with the description in [MMB+12] (sec. 2), PROV is defined by the following core relations.

$$\begin{aligned} used &\subseteq Act \times En \\ genBy &\subseteq En \times Act \\ wasDerivedFrom &\subseteq En \times En \\ inval &\subseteq En \times Act \\ waw &\subseteq Act \times Ag \\ abo &\subseteq Ag \times Ag \\ wat &\subseteq En \times Ag \\ wasInformedBy &\subseteq Act \times Act \end{aligned}$$

These are summarized in Fig. 3. Initially, we are going to restrict ourselves to an even simpler model, consisting only of En, Act, and relations $used$ and $genBy$. Agents and the relations that involve them are introduced in Sec. 5. Further extensions to the additional relations — $wasDerivedFrom$ and $wasInformedBy$ — are straightforward and are not considered in detail due to space constraints.

An instance I of the model consists of sets $en \in En$ and $act \in Act$ of symbols, and sets of relation instances $\{genBy(e,a) \mid e \in En, a \in Act\} \cup \{used(a,e) \mid e \in En, a \in Act\}$.

As these relations are binary, we can view I as a digraph $G = (V, E)$, where $V = En \cup Act$, and each relation instance maps to a labelled directed edge. By convention, we orient these edges from right to left, to denote that the relation "points back to the past". Thus: $a \xleftarrow{genBy} e \in E$ iff $genBy(e,a) \in I$, and $e \xleftarrow{used} a \in E$ iff $used(a,e) \in I$. We denote the label associated to edge (v_i, v_j) as $label(v_i, v_j)$.

Note that, by definition of the relations, G is a bipartite graph. We denote the set of all such graphs by $PG_{gu/ea}$, to indicate that they only contain En and Act nodes, and $genBy$ and $used$ edges. In Sec. 5 we are going to extend this set to include agents as well as additional relations. Fig. 4 portrays a simple $PG_{gu/ea}$ graph that we will be using as a running example.

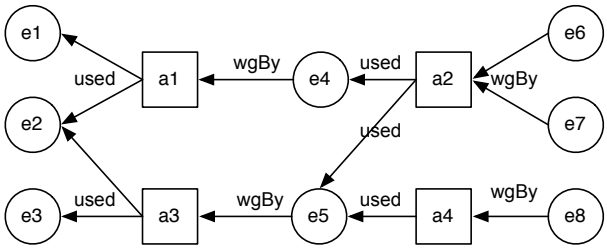

Fig. 4. $PG_{gu/ea}$ provenance graph used as a running example to illustrate abstraction by grouping

3.1 $PG_{gu/ea}$ constraints

The PROV-CONSTR specification document [CMM12] defines two main types of constraints, namely type constraints and temporal constraints. The former specify the domain and range of relationships amongst entities, activities, and agents. For instance, "*a* used *e*" is a valid relation if and only if *e* is an entity node and *a* is an activity node. These constraints are listed in [CMM12] (see Sec. 6.3, **Constraint 50 (typing)**) and are summarized by the schema definitions above. Additionally, constraint **entity-activity-disjoint (Constraint 55)** in [CMM12] stipulates that entities and activities be disjoint: $En \cap Act = \emptyset$..

Temporal constraints rely on a set of pre-defined temporal events, which are used to mark the start and end of an activity, the start of entity usage, the end of entity generation, and more. An example of constraints in this class is the following: "given two events ev_1, ev_2 which mark the start and end of *a*, respectively, then any event *ev* which marks the usage of *e* by *a* cannot temporally precede ev_1 or follow ev_2."

A *valid* PROV graph is one that satisfies all the applicable constraints. Although both these types of constraints are an integral part of the definition of validity of a provenance graph, in this work we are going to focus exclusively on type constraints to define validity. The implications of the grouping operator on temporal events, and thus the preservation of the corresponding constraints, is the subject of related but separate work.

4 Grouping Provenance Graph Nodes

As mentioned in Sec. 1.2, our goal is to define graph editing operators that selectively remove information from a graph $G \in PG_{gu/ea}$, yielding a new graph $G' \in PG_{gu/ea}$. The first of these, namely the removal of labels or annotations associated with a node or an edge, is straightforward. Regarding the removal of a node, we note that simply reconnecting the remaining nodes generally may lead to an invalid graph. A simple example is a graph defined by: $\{used(a, e_1), genBy(e_2, a)\}$

where activity a is removed. This results simply in two disconnected nodes e_1, e_2, because no relationship can be inferred between them from the original graph.

Rather than delving into the possible consequences of such node and edge elision, we are going to focus exclusively on the $Group$ graph transformation operator as the prime way to achieve abstraction over provenance graphs. $Group$ takes a graph $G = (V, E) \in PG_{gu/ea}$ and a subset $V_{gr} \subset V$ of its nodes and produces a modified graph $G' \in PG_{gu/ea}$. The nodes in V_{gr} are "grouped" together and replaced by a new single node.

$$Group : PG_{gu/ea} \times \mathbb{P}(V) \to PG_{gu/ea}$$

As the operator is closed under composition, further abstraction can be achieved by repeated grouping, either on multiple disjoint sets V_{gr}, or on sets that include abstract nodes (abstraction of abstraction).

To get a quick intuition of the problems faced in the definition of the grouping operator, consider the transformation in Fig. 5, where nodes $V_{gr} = \{e_1, e_3, e_4, e_5\}$ are simply replaced with new node e' in the example graph of Fig. 4, and all edges in and out of nodes in V_{gr} are just "rewired" in and out of e'.

This naive replacement leads to problems. Firstly, it introduces two cycles: $e' \leftrightarrow a_1$ and $e' \leftrightarrow a_3$. Furthermore, the two edges $e' \leftarrow a_1$ and $e' \leftarrow a_3$ cannot be of type $genBy$, while at the same time one cannot arbitrarily introduce $used$ relations, which would be false dependencies. Thus, the resulting graph is not a valid PROV graph. Note that the former of these problems had been already pointed out in the description of the ProPub system [DZL11], mentioned above.

4.1 Closure and homogeneous grouping

The example suggests that the cycle is caused by nodes a_1 and a_3, which both lie on the paths between two of the nodes in V_{gr}. Intuitively, set V_{gr} is not "convex", that is, there are paths in G that lead out of V_{gr} and then back in again. This observation suggests the introduction of a preliminary *closure* operation, aimed at ensuring acyclicity. This is defined as follows.

Definition 1 (Path Closure): Let $G = (V, E) \in PG_{gu/ea}$ be a provenance graph, and let $V_{gr} \subset V$. For each pair $v_i, v_j \in V_{gr}$ such that there is a directed path $v_i \rightsquigarrow v_j$ in G, let $V_{ij} \subset V$ be the set of all nodes in the path. The Path Closure of V_{gr} in G is

$$pclos(V_{gr}, V) = \bigcup_{v_i, v_j \in V_{gr}} V_{ij}$$

Fig. 6(b) shows a continuation of the previous example. This time the replacement is performed on $pclos(\{e_1, e_3, e_4, e_5\}, G) = \{e_1, e_3, e_4, e_5, a_1, a_3\}$, resulting in graph (c). However, while this solves the cycle problem, the graph is no longer bipartite, due to the new edges $e' \to e_2$ and $e' \to e_6$, which connect nodes of the same type. In this example, we can construct a new group of nodes, $\{e', e_2, e_6\}$, on the graph that results from the first replacement, and replace it with a new node e''. The resulting graph (d) is a valid $PG_{gu/ea}$ graph.

The same result can be obtained by first *extending* the closure in (b) to include e-nodes e_2, e_6, and then replacing the resulting set with e'' (this is indicated by the "extend and replace" arrow from (b) to (d) shown in the figure). Following this approach, we are going to define grouping as a composition of three functions: *closure*, defined above, *extension*, and *replacement*, as follows.

The *extension* of a set $V_{gr} \subset V$ relative to type $t \in \{En, Act\}$ is V_{gr} augmented with all its adjacent nodes, in either direction, of type t. Formally:

Definition 2 (extend): Let $G = (V, E) \in PG_{gu/ea}$, $t \in \{En, Act\}$.

$$\begin{aligned} &extend(V_{gr}, G, t) = \\ &\quad V_{gr} \cup \\ &\quad \{v' \mid (v, v') \in E \wedge v \in V_{gr} \wedge type(v') = t)\} \cup \\ &\quad \{v \mid (v', v) \in E \wedge v \in V_{gr} \wedge type(v') = t)\} \end{aligned}$$

In our example, $extend(\{e_1, e_3, e_4, e_5, a_1, a_3\}, G, En) = \{e_1, e_3, e_4, e_5, a_1, a_3, e_2, e_6\}$. Note that all sink nodes in $extend(V_{gr}, G, t)$ are of type t by construction.

Next, we consider replacement. Let $G = (V, E)$, $V'_{gr} \subset V$ be obtained using $extend$, and let v_{new} be a new node symbol that does not appear in V. Function $replace$ replaces V' with v_{new} in V, and connects v_{new} to the rest of the graph, as follows.

Let $\vartheta_{out}(V'_{gr})$ denote the *outcut* of G associated with V'_{gr}:

$$\vartheta_{out}(V'_{gr}) = \{(v, v') \mid v \in V_{gr'}, v' \in V \setminus V_{gr}\}$$

This is the set of arcs of G leading out of V'_{gr}, i.e., whose heads lie in V'_{gr} and whose tails lie in $V \setminus V'_{gr}$. Symmetrically, let $\vartheta_{in}(V'_{gr})$ denote the *incut* of G associated with V'_{gr}, i.e., the set of arcs of G leading into V'_{gr}:

$$\vartheta_{in}(V'_{gr}) = \{(v, v') \mid v' \in V_{gr'}, v \in V \setminus V_{gr}\}$$

Finally, let $\vartheta_{int}(V'_{gr}) = \{(v_1, v_2) \in E \mid v_1, v_2 \in V'_{gr}\}$ denote the arcs that connect nodes in V'_{gr}:

$$\vartheta_{int}(V'_{gr}) = \{(v, v') \mid v, v' \in V_{gr'}\}$$

Function $replace$ replaces each arc $(v', v) \in \vartheta_{out}(V'_{gr})$ with a new arc (v_{new}, v) of the same type, and replaces each arc $(v, v') \in \vartheta_{in}(V'_{gr})$ with a new arc (v, v_{new}) of the same type. Arcs in $\vartheta_{int}(V'_{gr})$ simply disappear along with the nodes in V'_{gr}. It is easy to verify that the resulting graph is type-correct. Indeed, all sink nodes in V'_{gr} are of type t as noted above, and so is v_{new} by construction. Thus, sink nodes are replaced by a node v_{new} of the same type. Since the arcs have the same type as those they replace, it follows that $replace$ preserves type correctness. Formally, let:

Definition 3:

$$\vartheta'_{out}(V'_{gr}) = \{v \xleftarrow{t} v_{new} \mid v \xleftarrow{t} v' \in \vartheta_{out}(V'_{gr})\}$$

$$\vartheta'_{in}(V'_{gr}) = \{v_{new} \xleftarrow{t} v \mid v' \xleftarrow{t} v \in \vartheta_{in}(V'_{gr})\}$$

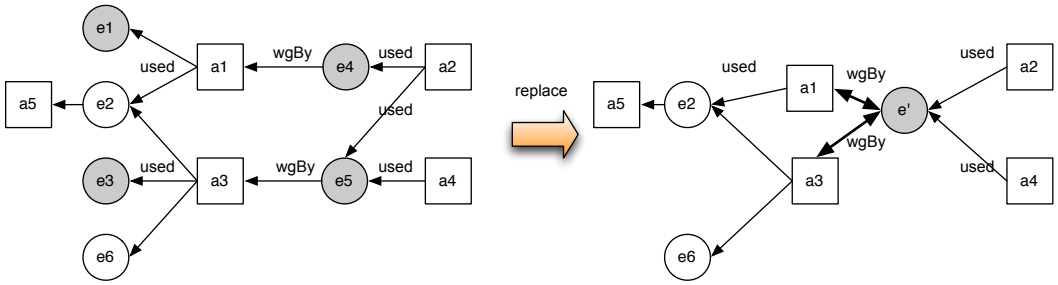

Fig. 5. A naive replacement of a set of nodes may lead to a non-$PG_{gu/ea}$ graph.

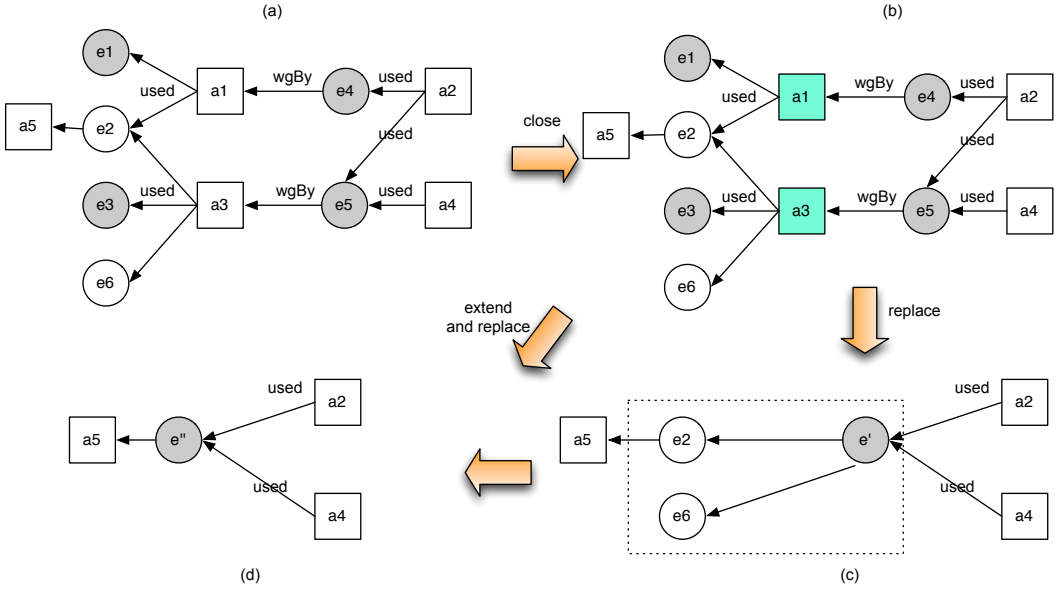

Fig. 6. Path Closure and replacement with extension on a set of Entity nodes.

Definition 4 (Replace):

$$replace(V_{gr}, v_{new}, G) = (V', E'), \text{ where:}$$

$$\begin{aligned} V' &= V \setminus V_{gr} \cup \{v_{new}\} \\ E' &= E \setminus (\vartheta_{out}(V_{gr}) \cup \vartheta_{in}(V_{gr}) \cup \vartheta_{int}(V_{gr})) \\ &\quad \cup \vartheta'_{out}(V_{gr}) \cup \vartheta'_{in}(V_{gr}) \end{aligned}$$

We can now provide an initial definition of our $Group$ operator, under the simplifying assumption that all nodes in V_{gr} are of the same type, either En or Act, which we denote by $type(V_{gr})$ (with a slight abuse of notation). Fig. 7 shows a progression similar to that of Fig. 6, but this time $type(v) = Act$ for each $v \in V_{gr} = \{a_1, a_2, a_3\}$, and V_{gr} is replaced by another activity node, a'. Under assumption of type homogeneity, the grouping operator is a functional composition of $pclos$, $extend$, and $replace$ functions, defined as follows.

Definition 5 (Homogeneous Grouping): Let $G = (V, E) \in PG_{gu/ea}$, $V_{gr} \in V$ be a type-homogeneous set, and let v_{new} be a new node with $type(v_{new} = type(V_{gr})$.

$$\begin{aligned} &Group_{hom}(G, V_{gr}, v_{new}) = \\ &\quad replace(\\ &\quad extend(\\ &\quad pclos(V_{gr}, V), V, type(V_{gr})), v_{new}, G) \end{aligned}$$

As an illustration, in our running example we have:

$$\begin{aligned} V_{gr} &= \{e_1, e_3, e_4, e_5\} \\ V_{cl} &= pclos(V_{gr}, G) = \{e_1, e_3, e_4, e_5, a_1, a_3\} \\ V' &= extend(V_{cl}, En) = V_{cl} \cup \{e_2, e_6\} \\ Group_e(G, V_{gr}, v_{new}) &= replace(V', v_{new}, G) \\ &= (\{e''\}, \{(a_2, e''), (a_4, e'')\}) \end{aligned}$$

4.2 Generalization to *e-grouping* and *a-grouping*

So far we have considered grouping over sets V_{gr} of type-homogeneous nodes (before closure). Additional care must be taken if we allow V_{gr} to include nodes of mixed types. First, the type of the replacement node must now be specified, as it is no longer implied from the type of

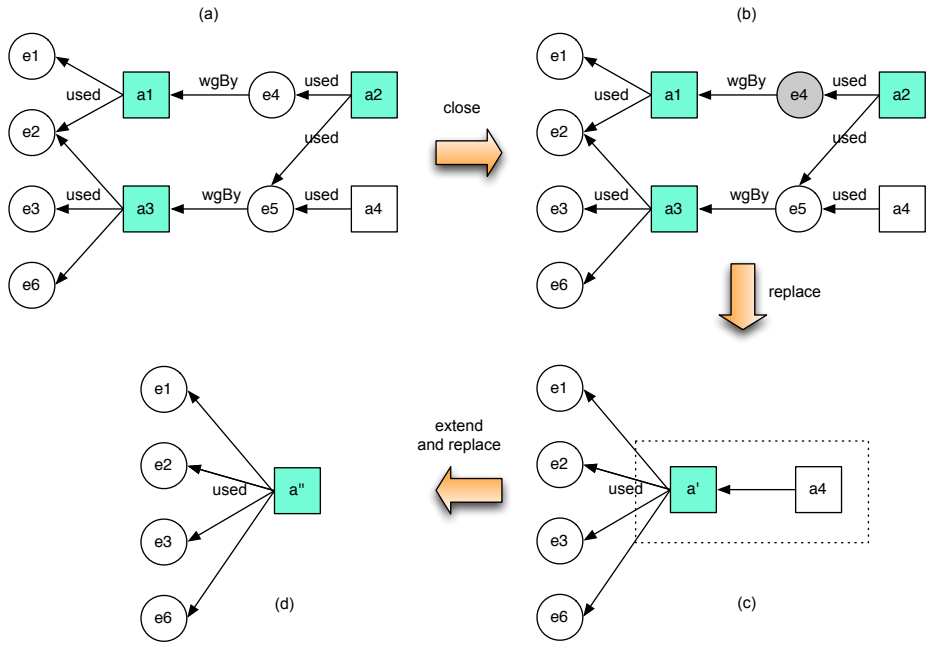

Fig. 7. Grouping on a set of Activity nodes

the nodes in V_{gr}. Indeed, the choice of such type leads to different abstracted graphs. Thus, we will now refer to grouping as *t-grouping*, where $t \in \{En, Act\}$, i.e., **e-grouping** or **a-grouping**. Fig. 8(a-1, a-2) illustrates the application of the $Group_{hom}$ operator (Def. 5), assuming *a-grouping* and $V_{gr} = \{e_4, a_2\}$. Note that the extension incorporates Activity node a_1.

Second, observe that a new pattern arises in the case of *e-grouping* as shown in Fig. 8(e-1, e-2). Now the extension leads to $V_{cl} = V_{gr} \cup \{e_5\}$, which in turn leads to the pattern shown in Fig. 8(e-3), involving two generation events for the new entity e_N.

Although this is a valid pattern, the two generation events must be simultaneous (this is one of the temporal constraints defined in [CMM12]):

$$ev(genBy(e_N, a_1)) \preceq ev(genBy(e_N, a_2)) \quad \wedge$$
$$ev(genBy(e_N, a_2)) \preceq ev(genBy(e_N, a_1))$$

The intuitive interpretation for this pattern is that each of the two activities generated one entity in the group represented by e_N, and that the abstraction makes these two events indistinguishable. Formally, nothing further needs to be done to the graph. However note that one can restore, if desired, the more natural pattern whereby one single generation event is recorded for e_N. This is achieved simply by propagating the grouping to the set of generating activities. In the example, this leads to the graph in Fig. 8(e-3).

We now formalize these considerations by introducing two definitions for $Group$. The first, which we call **t-grouping**, is agnostic of multiple generation patterns, while the second (**strict t-grouping**) applies propagation to ensure that the graph is free from multiple generation patterns.

Definition 6 (t-Grouping): Let $G = (V, E) \in PG_{gu/ea}$, $V_{gr} \in V$, $t \in \{En, Act\}$, and let v_{new} be a new node with $type(v_{new}) = t$. Then:

$$Group(G, V_{gr}, v_{new}, t) = replace(extend(pclos(V_{gr}, V), V, t), v_{new}, G)$$

Note that the assumption that sink nodes in the closure are homogeneous and are replaced by a node of the same type t, which is necessary for $replace$ to perform correctly, still holds in this case.

Definition 7 (Strict t-Grouping): Given $G = (V, E) \in PG_{gu/ea}$, $V_{gr} \in V$, $t \in \{En, Act\}$, and a new node v_{new} with $type(v_{new}) = t$, let

$$G' = (V', E') = Group(G, V_{gr}, v_{new}, t).$$

Let $V_{gen} = \{a \in V' \mid a \xleftarrow{genBy} v_{new} \in E'\}$ be the set of Activity nodes that generate v_{new} according to G', and let a_{new} be a new activity node. Then:

$$Group_{str}(G, V_{gr}, v_{new}, t) = \begin{cases} G' & \text{if } |V_{gen}| \leq 1 \\ replace(V_{gen}, a_{new}, G') & \text{otherwise} \end{cases}$$

It is straightforward to show that strict t-grouping reduces to normal grouping if the grouping is a homogeneous a-grouping:

if $type(t) = Act$
then $Group_{str}(G, V_{gr}, t) = Group(G, V_{gr}, t)$.

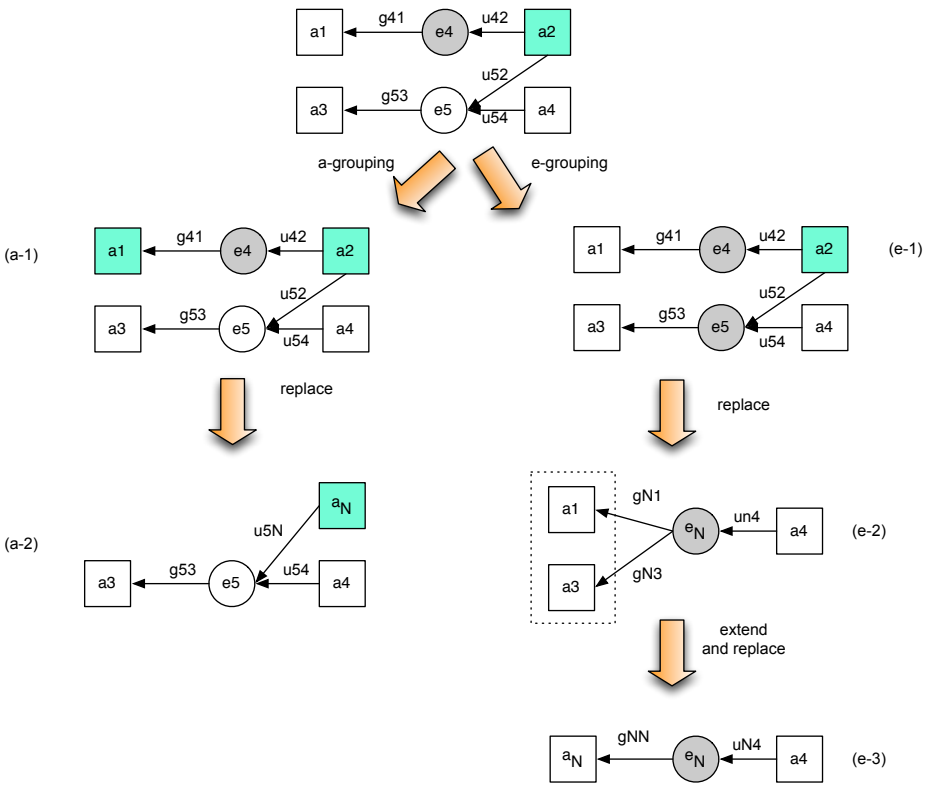

Fig. 8. e-grouping and a-grouping on mixed type nodes

5 Abstraction with Agents

Having laid the foundations for abstraction on the core $PG_{gu/ea}$ model, extending grouping to a model that also includes agents, the third pillar or the PROV model, is quite straightforward. Agents may be humans or software systems. Specifically, we now consider the node type Ag and the following additional relation types from the PROV schema of Sec. 3:

$$\begin{aligned} waw &\subseteq Act \times Ag \\ wat &\subseteq En \times Ag \\ abo &\subseteq Ag \times Ag \end{aligned}$$

We use the shorthand relation names waw, wat and abo for *wasAssociatedWith*, *wasAttributedTo* and *actedOnBehalfOf*. These denote responsibility of an agent for an activity (waw), responsibility of an agent for an entity (wat), and delegation between two agents (abo).

Note that PROV admits an additional optional activity element to abo, which is used to qualify the delegation as occurring within the scope of that activity. For simplicity, we are not going to consider this qualified version of the relation. Thus, we can still assume that these new relations are binary, and so we continue to view an instance of a provenance graph as a digraph $G = (V, E)$, where new $V = En \cup Act \cup Ag$, and where each relation instance maps to a labelled directed edge. We denote the set of all such graphs as $PG_{gu+/eaAg}$.

The main implications of adding agents to our abstraction model are that (i) a new *ag-grouping* operator must be introduced, and (ii) the existing definitions of e-grouping and a-grouping must be modified slightly.

In order to incorporate agents into the definition of $Group$, observe that, for all three relations involving agents, the agent node is always the *target* of the directed edge. This means that agents can be viewed as part of an "outer layer" in the provenance graph. This is illustrated in Fig. 9, where the agents are shown in bold lines in the outer side of the digraph.

This observation suggests we can break down the analysis of grouping with agents into the following three parts.

1) $V_{gr} \subset Ag$. This is the case for ag-grouping, which only involves the outer layer of the graph. Since agents are only related to each other through delegation: $abo(ag_1, ag_2)$, grouping in this case is akin to *homogeneous grouping* from Sec. 4.1, i.e., no nodes of other types are ever involved, and $v_{new} \in Ag$.
2) $V_{gr} \subset En \cup Act$ as in Sec. 4. This is the case of t-grouping (Def. 6), where the nodes involved in the abstraction are in the inner layer, but they may be related to agent nodes via waw and wat relations.
3) $V_{gr} \subset En \cup Act \cup Ag$. Here, the group set may contain any combination of nodes. However, the peripheral role played by agents relative to entities and activities suggests that it may be reasonable to restrict this case to e-grouping or a-grouping, i.e., a

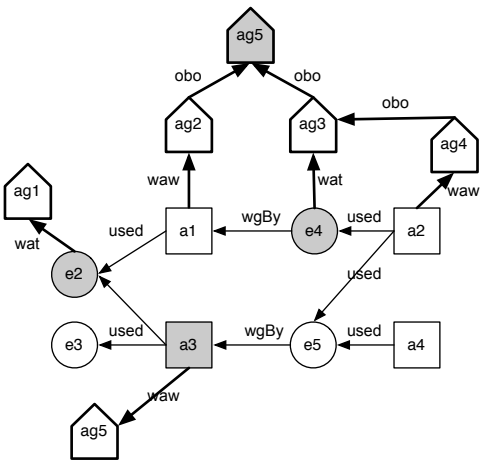

Fig. 9. $PG_{gu+/eaAg}$ provenance graph. Agents lie on the outer border of the digraph. Shaded nodes show a possible grouping set in the general case.

combination of node types may include agents, but it should not be abstracted by a new agent node.

5.1 Ag-grouping: abstracting agents

We begin with the case where abstraction is performed over a set of agents, i.e., $V_{gr} \subset Ag$. In this case, the existing definition of t-grouping (Def. 6) extends naturally to $PG_{gu+/eaAg}$ graphs. T-grouping involves three operators: $pclos$, $extend$, and $replace$. Since $pclos(V_{gr}, V)$ operates only on abo relations, it follows that its result is also homogeneous, i.e., $pclos(V_{gr}, V) \subset Ag$. Also, there is no need to restore type validity by extending the closure, i.e., $extend$ is the identity: $extend(pclos(V_{gr}, V), V, Ag) = pclos(V_{gr}, V)$. Finally, it is easy to see that our original definition of group replace (Def. 4) is general enough to accommodate the "rewiring" of the new abstract agent node. We illustrate this informally using the patterns of Fig. 10.

In pattern (a), $V_{gr} = \{ag_1, ag_4\}$, and $pclos(V_{gr}, V) = \{ag_1, ag_2, ag_3, ag_4\}$. In this case, the closure includes all the intermediate agents in the delegation chain between ag_1 and ag_4. Replacement trivially transforms G into the single abstract agent ag_N.

In pattern (b), $V_{gr} = \{ag_2, ag_5, ag_4\}$. Note that not all agent nodes in V_{gr} are related, either directly or through a path. This is not a problem, as we have $V_{clos} = pclos(V_{gr}, V) = \{ag_2, ag_5, ag_4, ag_3\}$. Replacement applies as follows, where $\vartheta_{int}(V_{clos})$ is the set of relations beginning and ending inside V_{clos}:

$$\begin{aligned} \vartheta_{int}(V_{clos}) &= \{abo(ag_2, ag_3), abo(ag_3, ag_5)\} \\ \vartheta_{in}(V_{clos}) &= \{wat(e, ag_2), abo(ag_1, ag_4)\} \\ \vartheta_{out}(V_{clos}) &= \{abo(ag_4, ag_6)\} \end{aligned}$$

Thus, $replace(V_{clos}, ag_N, G)$ maps relations in the original graph to those in the abstracted graph as follows:

$$\begin{aligned} abo(ag_4, ag_6) &\rightarrow abo(ag_N, ag_6) \\ wat(e, ag_2) &\rightarrow wat(e, ag_N) \\ abo(ag_1, ag_4) &\rightarrow abo(ag_1, ag_N) \end{aligned}$$

In practice, replacement preserves agents ag_1 and ag_6 and restores their delegation relations relative to the new abstract agent, ag_N. It also maps relation $wat(e, ag_2)$, which involves the untouched e node, to a new relation of the same type: $wat(e, ag_N)$.

We conclude that, in this first case, Def. 5 applies without changes.

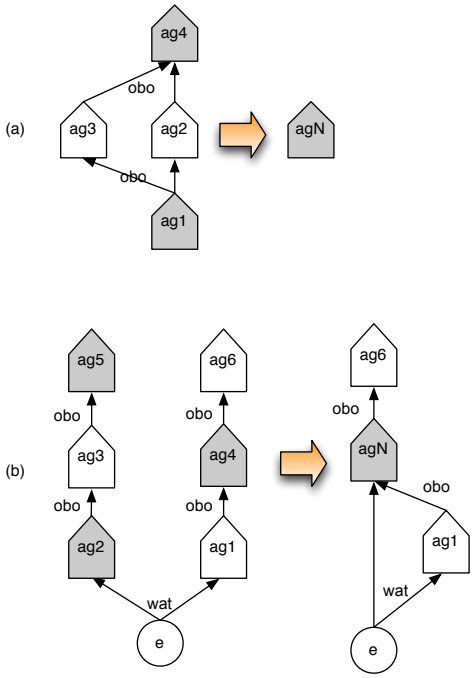

Fig. 10. ag-grouping involving delegation.

5.2 a-grouping and e-grouping with agents

The second case, where $V_{gr} \subset En \cup Act$, is t-grouping with added agents relations. Here the $replace$ operator (Def. 4) must now consider how edges that involve agents are mapped to new edges in the abstract graph. We have already observed that agent nodes are always the targets of directed edges. It follows that the closure of a set of nodes $V_{gr} \subset En \cup Act$ never adds agent nodes to V_{gr}, because this would require the added agent to be on a path between two nodes from $En \cup Act$, and therefore to be the source of a directed edge.

A second observation is that if $waw(a, ag)$ holds, and a is involved in a-grouping, then a is replaced by a_{new}, and thus $waw(a_{new}, ag)$ also holds (Fig. 11(a1)). Similarly for e-grouping, if $wat(e, ag)$ holds, and e is involved in e-grouping, then e is replaced by e_{new}, and $wat(e_{new}, ag)$ holds (Fig. 11(b1)). On the other hand, suppose $waw(a, ag)$ holds and e-grouping is performed.

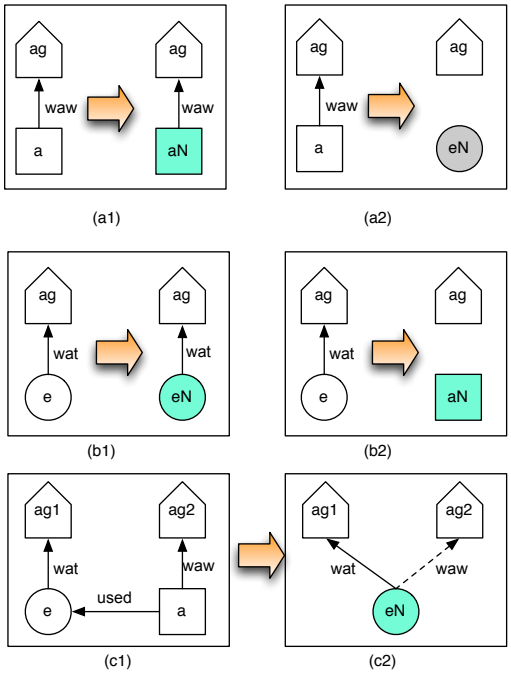

Fig. 11. waw and wat edges involving nodes in $pclos(V_{gr}, V)$ may need to removed following e-grouping and a-grouping, respectively.

A simple case is shown in Fig. 11(c1). In this case, a is replaced by $e_{new} \in En$, therefore $waw(e_{new}, ag)$ is type-incorrect. Similarly, $wat(e, ag)$ after a-grouping would become, incorrectly, $wat(a_{new}, ag)$. These two patterns are summarised in Fig. 11(a2, b2 resp.). Note that one cannot simply replace association with attribution, i.e., replace relation $waw(a, ag)$ with $wat(e_N, ag)$, because there is no guarantee that any of the entities represented by the new e_N had been attributed to ag in the original graph. Similarly, one cannot replace $wat(e, ag)$ with $waw(a_N, ag)$. Instead, in pattern (a) we simply remove the incorrect waw relations following e-grouping, and similarly, in pattern (b) we remove the incorrect wat relations following a-grouping.

These considerations suggest that the definition of the $replace$ function for grouping needs to be adapted for the case where agents are involved. To understand why, recall from Sec. 4.1 that $replace$ replaces a type-homogeneous set, computed by the $extend$ function, with an abstract node of the same type. An extension of type t augments the closure of a grouping set by adding all adjacent nodes of the same type to it. This ensures that replacing the nodes in the extension with an abstract node of the same type preserves the type correctness of the relations. However it should be clear from the example above (parts c1, c2 of the figure), that both e-grouping and a-grouping on the set $V_{gr} = \{a, e\}$ result in one of the two agent relations being incorrect. Intuitively, this is because the extension function fails to incorporate agents, leaving the agent relations exposed

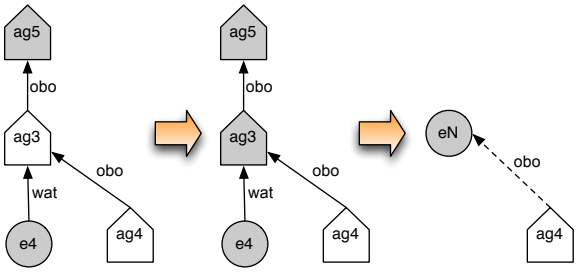

Fig. 12. e-grouping leads to incorrect abo relation when agents are part of a closure.

on the outcut of V_{gr} (in fact, $extend$ in this example does nothing at all).

The pattern in (c2) (or its symmetric, for a-grouping), can be obtained simply by ensuring that $replace$ deletes the incorrect relations. The following variation on Definition 3 ensures these deletions are enforced.

$$\vartheta'_{out}(V'_{gr}) = \{v \xleftarrow{t} v_{new} \mid v \xleftarrow{t} v' \in \vartheta_{out}(V'_{gr}) \wedge ((t = wat \wedge type(v_{new}) = En) \vee (t = waw \wedge type(v_{new}) = Act) \vee (t \neq wat \wedge t \neq waw))\}$$

5.3 The general case: grouping on any node type

The third and more general case, where $V_{gr} \subset En \cup Act \cup Ag$, presents one further difficulty. Consider performing e-grouping on the pattern of Fig. 12 (left), with $V_{gr} = \{e_4, ag_5\}$. We have $pclos(V_{gr}) = \{e_4, ag_5, ag_3\}$, resulting in the abstraction on the right. Clearly, the former $abo(ag_4, ag_3)$ relation should not be mapped in the final graph. This issue arises because there the extension function, which guarantees type consistency for e- and a-nodes, does not include agents.

Once again we deal with the issue by changing the definition of $replace$ to ensure that the incorrect relation is not mapped. Note that a change is required to $\vartheta'_{in}(V'_{gr})$ rather than $\vartheta'_{out}(V'_{gr})$ as in the previous case. The new definition is as follows.

$$\vartheta'_{in}(V'_{gr}) = \{v_{new} \xleftarrow{t} v \mid v' \xleftarrow{t} v \in \vartheta_{in}(V'_{gr}) \wedge ((t = abo \wedge type(v_{new}) = Ag) \vee t \neq abo)\}$$

Thus, a delegation relation is mapped in the abstract graph only if the target abstract node is an agent.

With the new version of the $replace$ function, introduced in the previous two sections, some of the relations in the original graph G are not mapped to the abstracted graph G'. This makes it possible for some of the nodes in G' to end up disconnected from the rest of the graph. As an example, the graph in Fig. 13 shows the result of e-grouping over the shaded nodes in the graph of Fig. 9. The combination of closure, extensions and replacement results in ag_5 being isolated, or "orphaned" in the abstract graph.

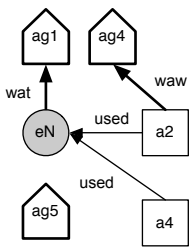

Fig. 13. Abstracted version of the graph in Fig. 9 after e-grouping involving agents.

As isolated agent nodes may not be significant to consumers of the abstracted graphs, for completeness we provide a simple function to optionally remove them at the end of the abstraction process, as follows.

Definition 8 (Removing isolated agents): Let $G = (V, E) \in PG_{gu+/eaAg}$, and

$$isolated(G) = \{v \in V \mid type(v) = Ag \\ \wedge \nexists v' \in V.((v', v) \in E \vee (v, v') \in E)\}$$

Then:

$$remIsolated((V, E)) = (V \setminus isolated(G), E)$$

6 Summary and Further Research

Rather than giving a procedural pseudo-code for the $Group$ algorithm, we summarize its functional specification, comprising of the four functions defined so far: $pclos()$, $extend()$, $replace()$, and $remIsolated()$, in Table 1.

A procedural implementation of the algorithm, written in Java, is available online (https://github.com/PaoloMissier/ProvAbs) as part of a tool in which a policy language is used to drive the selection of the set V_{gr} of nodes to be abstracted. Using the tool, policy setters may experiment with abstraction policies on their provenance graphs, observing their effects when abstraction is performed. Noting that $Group()$ is closed wrt composition, the tool also allows for more complex abstraction, by letting users specify compositions of $Group$ operations as part of their policy. The tool is described in more detail in a separate submission [MBG13]. An example of policy is given in the Appendix.

The work described in this paper is progressing in two main directions. First, we are aware that the fragment of PROV to which this version of $Group$ applies does not cover all relation types. Nevertheless, the method described in the paper for reasoning about PROV graph transformation can be used as a guideline to extend the work to the missing parts of PROV. We are going to address these in the future.

Second, so far we have ignored the implications of abstraction on the space of events that are used to characterize the semantics of PROV. As constraints over the relative ordering of events are defined in detail in the PROV-CONSTR document, there is however an obligation to extend the notion of validity of PROV graphs to include those constraints. Thus, grouping must be shown to be validity-preserving relative to those constraints as well.

Appendix

Overview of Policy Model and Tool for Abstraction

The tool alluded to in the previous section operates on provenance graphs written in the provenance notation PROV-N [MMCSR12].

Given $G \in PG_{gu+/eaAg}$, the tool lets users specify a grouping set V_{gr} by means of an *abstraction policy*. It operates in two steps. Firstly, path expressions and predicates are used to select a set of nodes, and to assign a numerical *sensitivity* value to them. For example, the following rule contains a path expression that binds variables `process` and `data` to activity and entity nodes a, e, respectively, such that $used(a, e)$ holds and e is any node that is reachable from node with id `d14`:

```
for all (process used data)
    where (data descendantOf d14))
      setSensitivity(data, 10)
```

The sensitivity value of the selected data nodes is set to 10. Here *descendantOf* is a built-in query predicate that returns all nodes reachable from a given start node.

As an another example, the predicate

```
for all (data wgb process)
    where (process.classification > "conf" in classifications)
      setSensitivity(data, 9)
```

binds variables `data` and `process` to pairs of nodes d, p such that $genBy(d, p)$ and where the classification value $p.classification$ of p (a property of the node) is greater than "conf". This requires an ordered set of classification labels to be declared in a user-defined `classifications` list. During the first step, the tool evaluates the policy rules, resulting in the annotation of the selected nodes with sensitivity values.

This model operates on the same principle as the Bell-LaPadula security model. Each known receiver of a graph is pre-assigned a clearance level. This is determined by factors outside the scope of the tool, e.g. how much a receiver is trusted to the provenance owner. In the second step, node sensitivities are compared to the clearance level cl of the receiver, and V_{gr} is defined as the set of nodes whose sensitivity is lower than cl.

The rationale for these two steps is that sensitivity can be defined largely independently of the specific receiver, while the exact level of abstraction is relative to a receiver, who in this case is represented simply by a clearance level.

$$
\begin{aligned}
pclos(V_{gr}, V) &= \bigcup_{v_i, v_j \in V_{gr}} V_{ij} \\
extend(V_{gr}, G, t) &= V_{gr} \cup \{v' \mid (v, v') \in E \wedge v \in V_{gr} \wedge type(v') = t)\} \\
&\quad \cup \{v \mid (v', v) \in E \wedge v \in V_{gr} \wedge type(v') = t)\} \\
replace(V_{gr}, v_{new}, G) &= (V', E'), \text{where } \vartheta'_{out}(V'_{gr}) = \{v \xleftarrow{t} v_{new} \mid v \xleftarrow{t} v' \in \vartheta_{out}(V'_{gr}) \wedge ((t = wat \wedge type(v_{new}) = En) \vee \\
&\qquad (t = waw \wedge type(v_{new}) = Act) \vee \\
&\qquad (t \neq wat \wedge t \neq waw))\} \\
&\qquad \vartheta'_{in}(V'_{gr}) = \{v_{new} \xleftarrow{t} v \mid v' \xleftarrow{t} v \in \vartheta_{in}(V'_{gr}) \wedge ((t = abo \wedge type(v_{new}) = Ag) \vee t \neq abo)\} \\
V' &= V \setminus V_{gr} \cup \{v_{new}\} \\
E' &= E \setminus (\vartheta_{out}(V_{gr}) \cup \vartheta_{in}(V_{gr}) \cup \vartheta_{int}(V_{gr})) \cup \vartheta'_{out}(V_{gr}) \cup \vartheta'_{in}(V_{gr}) \\
remIsolated(V, E) \text{where} &= (V', E') \\
isolated(G) &= \{v \in V \mid type(v) = Ag \wedge \nexists v' \in V.((v', v) \in E \vee (v, v') \in E)\} \\
V' &= V \setminus isolated(G) \\
E' &= E \\
Group(G, V_{gr}, v_{new}, t) &= remIsolated(replace(extend(pclos(V_{gr}, V), V, t), v_{new}, G))
\end{aligned}
$$

TABLE 1
Functional summary of the grouping algorithm.

REFERENCES

[BBDH08] O Biton, S Cohen Boulakia, S B Davidson, and C S Hara. Querying and Managing Provenance through User Views in Scientific Workflows. In *ICDE*, pages 1072–1081, 2008.

[BCKS09] Smriti Bhagat, Graham Cormode, Balachander Krishnamurthy, and Divesh Srivastava. Class-based graph anonymization for social network data. *Proc. VLDB Endow.*, 2(1):766–777, August 2009.

[BFJM06] Jeremy Bryans, John S. Fitzgerald, Cliff B. Jones, and Igor Mozolevsky. Formal modelling of dynamic coalitions, with an application in chemical engineering. In *ISoLA*, pages 91–98. IEEE, 2006.

[BSS08] Uri Braun, Avraham Shinnar, and Margo Seltzer. Securing provenance. In *Proceedings of the 3rd conference on Hot topics in security*, pages 4:1—-4:5, Berkeley, CA, USA, 2008. USENIX Association.

[CKKT11a] Tyrone Cadenhead, Vaibhav Khadilkar, Murat Kantarcioglu, and Bhavani Thuraisingham. A language for provenance access control. In *Proceedings of the first ACM conference on Data and application security and privacy*, CODASPY '11, pages 133–144, New York, NY, USA, 2011. ACM.

[CKKT11b] Tyrone Cadenhead, Vaibhav Khadilkar, Murat Kantarcioglu, and Bhavani Thuraisingham. Transforming provenance using redaction. In *Proceedings of the 16th ACM symposium on Access control models and technologies*, SACMAT '11, pages 93–102, New York, NY, USA, 2011. ACM.

[CMM12] James Cheney, Paolo Missier, and Luc Moreau. Constraints of the Provenance Data Model. Technical report, 2012.

[DKM+11] Susan B Davidson, Sanjeev Khanna, Tova Milo, Debmalya Panigrahi, and Sudeepa Roy. Provenance views for module privacy. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '11, pages 175–186, New York, NY, USA, 2011. ACM.

[DKR+11] Susan B Davidson, Sanjeev Khanna, Sudeepa Roy, Julia Stoyanovich, Val Tannen, and Yi Chen. On provenance and privacy. In *Proceedings of the 14th International Conference on Database Theory*, ICDT '11, pages 3–10, New York, NY, USA, 2011. ACM.

[DKRB10] Susan B. Davidson, Sanjeev Khanna, Sudeepa Roy, and Sarah Cohen Boulakia. Privacy issues in scientific workflow provenance. In Paolo Missier, Vasa Curcin, and Susan Dadvidson, editors, *First International Workshop on Workflow Approaches to New Data-centric Science (WANDS'10)*, Indianapolis, 2010. ACM.

[DZL11] Saumen Dey, Daniel Zinn, and Bertram Ludäscher. ProPub: Towards a Declarative Approach for Publishing Customized, Policy-Aware Provenance. In Judith Bayard Cushing, James French, and Shawn Bowers, editors, *Scientific and Statistical Database Management*, volume 6809 of *Lecture Notes in Computer Science*, pages 225–243. Springer Berlin / Heidelberg, 2011.

[HSW07] Ragib Hasan, Radu Sion, and Marianne Winslett. Introducing secure provenance: problems and challenges. In *Proceedings of the 2007 ACM workshop on Storage security and survivability*, StorageSS '07, pages 13–18, New York, NY, USA, 2007. ACM.

[LT08] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 93–106, New York, NY, USA, 2008. ACM.

[MBG13] P. Missier, J. Bryans, and C. Gamble. ProvAbs: a tool to specify policies for controlling the disclosure of sensitive provenance graphs. *(Submitted, demo paper)*, 2013.

[MCF+11] 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*, 7(21):743–756, 2011.

[MMB+12] Luc Moreau, Paolo Missier, Khalid Belhajjame, Reza B'Far, James Cheney, Sam Coppens, Stephen Cresswell, Yolanda Gil, Paul Groth, Graham Klyne, Timothy Lebo, Jim McCusker, Simon Miles, James Myers, Satya Sahoo, and Curt Tilmes. PROV-DM: The PROV Data Model. Technical report, World Wide Web Consortium, 2012.

[MMCSR12] Luc Moreau, Paolo Missier, James Cheney, and Stian Soiland-Reyes. PROV-N: The Provenance Notation. Technical report, 2012.

[NPS12] Dang Nguyen, Jaehong Park, and Ravi Sandhu. Dependency path patterns as the foundation of access control in provenance-aware systems. In *4th USENIX Workshop on the Theory and Practice of Provenance*, 2012.

[ZG08] Elena Zheleva and Lise Getoor. Preserving the Privacy of Sensitive Relationships in Graph Data. In Francesco Bonchi, Elena Ferrari, Bradley Malin, and Yücel Saygin, editors, *Privacy, Security, and Trust in KDD*, volume 4890 of *Lecture Notes in Computer Science*, pages 153–171. Springer Berlin / Heidelberg, 2008.

[ZPL08] Bin Zhou, Jian Pei, and WoShun Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *SIGKDD Explor. Newsl.*, 10(2):12–22, December 2008.